LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-695525

Dynamic Mesh Adaptation for Front Evolution Using Discontinuous Galerkin Based Weighted Condition Number Mesh Relaxation

P. T. Greene, S. P. Schofield, R. Nourgaliev

June 21, 2016

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Dynamic mesh adaptation for front evolution using discontinuous Galerkin based weighted condition number mesh relaxation

Patrick T. Greene, Samuel P. Schofield, Robert Nourgaliev

Lawrence Livermore National Laboratory, Design Physics Division, Livermore, CA, 94551, USA

June 22, 2016

LLNL-TN-695525

Abstract

A new mesh smoothing method designed to cluster mesh cells near a dynamically evolving interface is presented. The method is based on weighted condition number mesh relaxation with the weight function being computed from a level set representation of the interface. The weight function is expressed as a Taylor series based discontinuous Galerkin projection, which makes the computation of the derivatives of the weight function needed during the condition number optimization process a trivial matter. For cases when a level set is not available, a fast method for generating a low-order level set from discrete cell-centered fields, such as a volume fraction or index function, is provided. Results show that the low-order level set works equally well for the weight function as the actual level set. Meshes generated for a number of interface geometries are presented, including cases with multiple level sets. Dynamic cases for moving interfaces are presented to demonstrate the method's potential usefulness to arbitrary Lagrangian Eulerian (ALE) methods.

Keywords: Weighted mesh smoothing, Condition number mesh relaxation, r-refinement, Level set, Discontinuous Galerkin discretization, ALE method

1. Introduction

Mesh relaxation and adaptation is used to improve mesh quality and, ultimately, solution accuracy. This arises in static and dynamic applications. In static applications, a mesh relaxation algorithm is applied to an initial generated mesh to improve mesh quality and may be used to concentrate mesh zoning in areas such as multi-material interfaces or boundary layers likely to require enhanced accuracy to resolve local length scales. In dynamic applications, in particular in Arbitrary Lagrangian Eulerian (ALE) [1] methods, the mesh motion may defer from the material motion and is a powerful tool allowing analysts to prescribe objectives for the mesh motion including proximity to the Lagrangian mesh but improved element quality and smaller zoning in regions of interest [2]. This work develops an innovative algorithm for utilizing condition number mesh relaxation to improve mesh quality while enhancing mesh resolution at interfaces or at any location as prescribed with discrete weights.

Several variants of mesh relaxers have been developed. The most widespread method is *Equipotential* or *Winslow mesh smoothing* [3]. In Winslow smoothing, physical mesh line positions are regarded as a function of a parametric variable and the gradient of the mesh line positions with respect to that parametric coordinate are minimized in an iterative procedure. A weight can be naturally incorporated that effectively scales the Jacobian of the zone and draws mesh into higher weight regions according to the ratio of weights. However, due to the global behavior of the relaxer, Winslow smoothing can significantly disrupt local mesh regions that may be very regular, therefore degrading the local mesh quality.

An increasingly popular alternative method is the *condition number mesh relaxation method* developed by Knupp [4, 5, 6]. The method defines a condition number functional at each vertex that is a function of the geometry of all the cells that share that vertex[1]. By minimizing the condition number at all the vertices, the

[1]Knupp originally defined a family of mesh relaxation methods distinguished by the choice of matrix norm associated with the element Jacobian. Here we use the term *condition number relaxation* to denote Knupp's method as prescribed using the Frobenius norm. This is the most widely used and recommended version.

overall quality of the mesh increases. The inclusion of a barrier function in the objective functional ensures that the method cannot produce an invalid mesh from an initially valid mesh making the method extremely robust. The original method did not provide a process for the flow physics to influence the mesh relaxation process. Knupp et al. [2] addressed this issue with their reference Jacobian method. The method would use the Lagrangian motion of the mesh to compute a reference Jacobian, which could then be used to constrain the mesh relaxation.

Váchal and Maire [7] extended the original work of Knupp and derived a weighted condition number mesh relaxation method for general unstructured computational meshes. Similar to the work of Knupp et al. [2], their goal was to provide a link between the physics of the problem and the mesh relaxation. This was achieved by the introduction of a weight term in the condition number functional. By varying the value of the weight, different cell sizes could be obtained. If the weights are computed from flow variables, this provides a method for clustering mesh cells near areas of interest. A focus of their work is the proper discretization of the weighted condition number and they mention a number of methods for computing the derivatives of the weight function needed in the optimization process. Although methods for discrete derivatives were discussed, all the results shown were computed using analytical functions and their derivatives. Extensions to use discretely defined weights, for instance piecewise constant zone centered data, have not been demonstrated in general due to difficulties in reconstructing sufficiently smooth (C^2) representations of the original discontinuous weights [8].

In this paper, we present a new mesh relaxation method that is designed to cluster mesh cells near moving fronts and interfaces. The method is based on the weighted condition number relaxation of Váchal and Maire. The discretization issues discussed by Váchal and Maire are solved by using a discontinuous Galerkin representation for the weights. This allows the spatial derivatives of the weight function to be easily computed since the spatial dependency of the discontinuous Galerkin projection comes from the prescribed basis functions. The method can adapt to dynamic fronts by having the weights be a function of a level set representation of the front. For cases when a level set representation of the front is not available, we provide a method for generating a low-order level set from discrete cell-center values. This low-order level set can then be used in the weight function to obtain results very similar to the actual level set.

The remainder of this paper is structured as such: Section 2 provides an overview of weighted condition number relaxation and presents the new discontinuous Galerkin weights. Section 3 introduces the level set based weights and provides a number of example meshes. Section 4 discusses computing the weight function from discrete cell-centered values for cases when a level set is not available. Section 5 explores combining the level set with the methods used for the discrete value based weights, which can be used when the level set is only computed within a narrow band near the interface. Section 6 presents a method for prescribing a minimum cell area for the resulting mesh. The examples from previous sections all involve taking a uniform mesh and then clustering cells near a stationary interface. In Section 7, the discrete value based weights are utilized in two examples of dynamic interfaces. The paper finishes with the conclusions in Section 8.

2. Discontinuous Galerkin weighted condition number mesh relaxation

The new discontinuous Galerkin based weighted condition number mesh relaxation is based on the weighted condition number mesh relaxation of Váchal and Maire [7], which in turn is an extension of the condition number relaxation method of Knupp [4, 5, 6]. We will begin with an overview of the method from Váchal and Maire. This will be followed by the introduction of the discontinuous Galerkin projection of the weight function. At the end of the current Section, we will describe the convergence measure employed in this work.

2.1 Weighted condition number mesh relaxation

Condition number relaxation (CNR) works by defining a functional at each vertex of a mesh that represents the quality of the neighboring cells as a function of nodal positions and then minimizing the functionals for the entire mesh. The original formulation of the CNR method is designed to create cells with orthogonal edge vectors and unit aspect ratios. The inclusion of a weight function permits the clustering of mesh cells in areas of interest. The local functional is minimized using Newton's method in a Jacobi method fashion. That is to say, a new vertex position is computed by minimizing the local functional with all the neighboring vertices fixed. Once the new positions for all the vertices are computed, the entire mesh is updated at once.

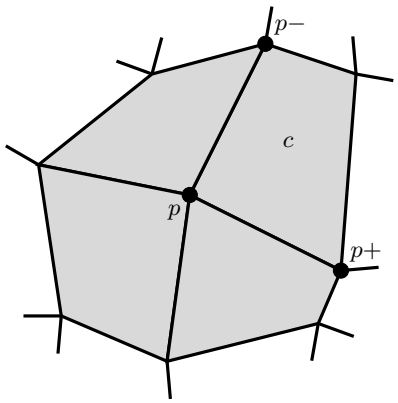
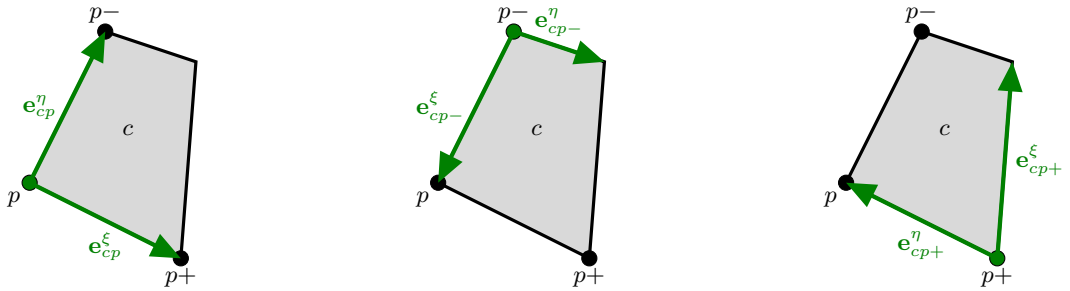


Figure 1: Sample mesh showing the cells affected by the minimization of the condition number for vertex p.



(a) Edge vectors for point p. (b) Edge vectors for point $p-$. (c) Edge vectors for point $p+$.

Figure 2: Edge vectors in cell c from Figure 1 used in the calculation of the functional for vertex p.

Only a single iteration of Newton's method is used at each vertex to find its new position. The process of applying a single step of Newton's method to each vertex and then updating all the vertex positions will be denoted as a single iteration of the mesh relaxation method.

Figure 1 shows a sample mesh around vertex p. In this work, we only consider quadrilateral cells, but the method can be extended to arbitrary cells. In addition, for simplicity, only two-dimensional meshes are discussed. The extension of the method to three-dimensional meshes is straightforward. All the cells affected by the movement of point p are shown in gray. Within a given cell, the vertex connected to p in the clockwise direction is labeled $p-$. Similarly, the counterclockwise vertex is labeled $p+$. Váchal and Maire [7] defined the local discrete functional for vertex p as:

$$F(\mathbf{x}_p) = \sum_{c \in \mathcal{C}(p)} \left(W^{cp} \frac{M^{cp}}{J^{cp}} + W^{cp-} \frac{M^{cp-}}{J^{cp-}} + W^{cp+} \frac{M^{cp+}}{J^{cp+}} \right), \tag{2.1}$$

where $\mathbf{x}_p = (x_p, y_p)$ is the position of vertex p, $\mathcal{C}(p)$ is the subset of all the cells that share p as a vertex, W^{cq} is the weight function computed in cell c at vertex q, and M^{cq} and J^{cq} are geometric functions computed from the edge vectors at vertex q of cell c. The functions are defined as

$$M^{cq} = \left\| \mathbf{e}^{\xi}_{cq} \right\|^2 + \left\| \mathbf{e}^{\eta}_{cq} \right\|^2 \qquad \text{and} \qquad J^{cq} = \left\| \mathbf{e}^{\xi}_{cq} \times \mathbf{e}^{\eta}_{cq} \right\|, \tag{2.2}$$

where $\| \cdot \|$ is the L_2 norm. $\mathbf{e}^{\xi}_{cq}$ and $\mathbf{e}^{\eta}_{cq}$ are the edge vectors starting at vertex q and pointing to the next vertex of cell c in the counterclockwise and clockwise direction, respectively. Figure 2 shows all the edge vectors in cell c from Figure 1 used to calculate the functional at vertex p.

The local discrete functional is minimized using Newton's method. This is equivalent to solving the

coupled equations

$$\frac{\partial F}{\partial x_p}(\mathbf{x}_p) = 0 \qquad \text{and} \qquad \frac{\partial F}{\partial y_p}(\mathbf{x}_p) = 0 \tag{2.3}$$

using the two-dimensional Newton-Raphson iterative algorithm. Solving these equations will require the first and second derivatives of M^{cq}, J^{cq}, and W^{cq}, with respect to x_p and y_p. The calculation of M^{cq}, J^{cq}, and their derivatives is straightforward since they only involve the edge vectors, which have an analytic form. Computing W^{cq} and its derivatives can be difficult as the weight may be a function of discrete values from the physics solver. For the Newton based minimization procedure, a reconstruction or smoothing step must be applied to the weights to obtain a sufficiently smooth approximation for the first and second derivatives of the weight function to be meaningful.

2.2 Discontinuous Galerkin weights

To create a twice differentiable representation of the weights within each cell, a piecewise-quadratic discontinuous Galerkin projection of the weight function is used. A piecewise-quadratic projection was chosen since it is the simplest projection with non-zero second derivatives. The projection allows the weight function within cell c to be expressed as

$$W^c(\mathbf{x}) = \sum_{n=1}^{N} W_n^c \, b_n^c(\mathbf{x}), \tag{2.4}$$

where W_n^c is the nth degree of freedom in cell c, b_n^c is the nth basis function in cell c, and N is the total number of degrees of freedom for the projection. For a two-dimensional piecewise-quadratic projection, there are a total of six degrees of freedom. Note, the projected weights are still discontinuous at element boundaries. When computing the quantity W^{cq}, we follow the advice of Váchal and Maire and use an average of the weight function evaluated at vertex q and its two neighbors ($q+$ and $q-$),

$$W^{cq} = \frac{1}{3}\Big(W^c(\mathbf{x}_q) + W^c(\mathbf{x}_{q+}) + W^c(\mathbf{x}_{q-})\Big). \tag{2.5}$$

By using a discontinuous Galerkin projection for the weights, the calculation of its derivatives becomes trivial, since they now only involve derivatives of the known basis functions. For example, the derivative of W^{cp} with respect to x_p can be expressed as

$$\frac{\partial W^{cp}}{\partial x_p} = \frac{1}{3}\frac{\partial}{\partial x_p}\Big(W^c(\mathbf{x}_p) + W^c(\mathbf{x}_{p+}) + W^c(\mathbf{x}_{p-})\Big). \tag{2.6}$$

In this expression, only the first term ($W^c(\mathbf{x}_p)$) is a function of x_p. Therefore, the derivative of the second and third terms ($W^c(\mathbf{x}_{p+})$ and $W^c(\mathbf{x}_{p-})$) are simply zero and Equation (2.6) simplifies to

$$\frac{\partial W^{cp}}{\partial x_p} = \frac{1}{3}\frac{\partial W^c}{\partial x_p}(\mathbf{x}_p). \tag{2.7}$$

Substituting in the discontinuous Galerkin projection, the expression for the derivative becomes

$$\frac{\partial W^{cp}}{\partial x_p} = \frac{1}{3}\sum_{n=1}^{N} W_n^c \frac{\partial b_n^c}{\partial x_p}(\mathbf{x}_p). \tag{2.8}$$

In the present work, we use the modal discontinuous Galerkin formulation [9]. The modal basis functions are derived from a Taylor series expansion around the centroid of the cell. The resulting basis functions for a two-dimensional piecewise-quadratic discontinuous Galerkin projection are

$$\begin{aligned} &b_1^c = 1, \quad b_2^c = \bar{x}, \quad b_3^c = \bar{y}, \\ &b_4^c = \frac{\bar{x}^2}{2} - \frac{1}{2\,\Omega_c}\int_{\Omega_c} \bar{x}^2 \, d\Omega, \quad b_5^c = \bar{x}\bar{y} - \frac{1}{\Omega_c}\int_{\Omega_c} \bar{x}\bar{y} \, d\Omega, \quad b_6^c = \frac{\bar{y}^2}{2} - \frac{1}{2\,\Omega_c}\int_{\Omega_c} \bar{y}^2 \, d\Omega, \end{aligned} \tag{2.9}$$

where

$$\bar{x} = \frac{x - x_c}{\Delta x}, \quad \bar{y} = \frac{y - y_c}{\Delta y}, \quad \Delta x = \frac{1}{2}\left(x_{\max} - x_{\min}\right), \quad \Delta y = \frac{1}{2}\left(y_{\max} - y_{\min}\right), \tag{2.10}$$

(x_c, y_c) is the centroid of the cell, Ω_c is the area of the cell, and $x_{\max}$, $x_{\min}$, $y_{\max}$ and $y_{\min}$ are the maximum and minimum nodal coordinates of the cell in the $x-$ and $y-$direction, respectively.

To permit the mesh to adapt to moving interfaces, the weight function will be computed from the signed distance function of a level set used to represent the interface. On its own, this can cause issues with the mesh because the signed distance function may only be C^0 continuous at some locations. If a discontinuity in the slope of the sign distance function is located within a cell, then the discontinuous Galerkin projection of the weight function can be ill behaved and lead to extreme stretching of the mesh. To address this, the degrees of freedom for the projection of the weights are computed using the weighted essentially non-oscillatory (WENO) reconstruction of Luo et al. [10, 11]. The WENO reconstruction provides a method for limiting the quadratic degrees of freedom and thus alleviating the issues caused by the discontinuities. WENO limiters are preferable for a Newton's method based algorithm because the solution will remain differentiable through the Newton iterations. Other limiters usually contain conditional statements in their algorithm. If a solution is near the conditional statement switch, Newton's method may oscillate between the two states and prevent convergence.

The WENO reconstruction of Luo et al. can be divided into three steps. First, the degrees of freedom for the constant and linear basis functions are computed using an L_2 projection of the weight function onto the basis functions. The resulting degrees of freedom provide a linear projection of the weight function, which is usually denoted as $\mathrm{DG(P_1)}$ in the literature. The second step of the method is to compute a least-squares reconstruction of the quadratic degrees of freedom from the constant and linear degrees of freedom in the current cell and its face neighbors. This provides a piecewise-quadratic projection of the weight function and the reconstructed projection is expressed as $\mathrm{rDG(P_1P_2)}$ in the literature [12, 13, 10, 11]. This projection will be ill behaved if any discontinuities are present in the cell. To rectify this, the final step of the WENO reconstruction is to apply a limiter to the quadratic degrees of freedom. The limiter is simply a weighted average of the degrees of freedom computed during the reconstruction. This will result in a new piecewise-quadratic projection which can then be used to compute the weights for the condition number functional. For completeness, the details of the method are provided in Appendix A.

The WENO reconstruction helps reduce the affect of any discontinuities found within a cell. Discontinuities, however, still exist at cell interfaces. This comes from the fact that the discontinuous Galerkin projection does not enforce continuity at the cell interfaces. This can result in different values being computed for a given vertex when projections from different cells sharing that vertex are used. In our work, there were no noticeable issues caused by these discontinuities and therefore no special treatment was used to correct or account for them.

2.3 Measure of convergence

Since the weighted condition number mesh relaxation method is an iterative method, a measure of convergence is required. In this work, the convergence is calculated as

$$\text{Convergence} = \frac{1}{N_{\mathcal{P}}} \sum_{p \in \mathcal{P}} \left(\frac{\|\Delta \mathbf{x}_p\|}{\sqrt{\Omega_R}} \right), \tag{2.11}$$

where $\Delta \mathbf{x}_p$ is the change in the position of vertex p during the last iteration, Ω_R is a reference area for the entire mesh, $\mathcal{P}$ is the set of all vertices in the computational domain, and $N_{\mathcal{P}}$ is the number of vertices in $\mathcal{P}$. For the reference area, we use the cell area of the initial uniform mesh.

3. Level set based weight function

3.1 Level set function and signed distance function

A level set is a function design to separate a computational domain into two regions. The level set will be negative within the first region and positive within the second region. The interface between the two regions is defined by the zero contour of the level set function. A specific type of level set is the signed distance

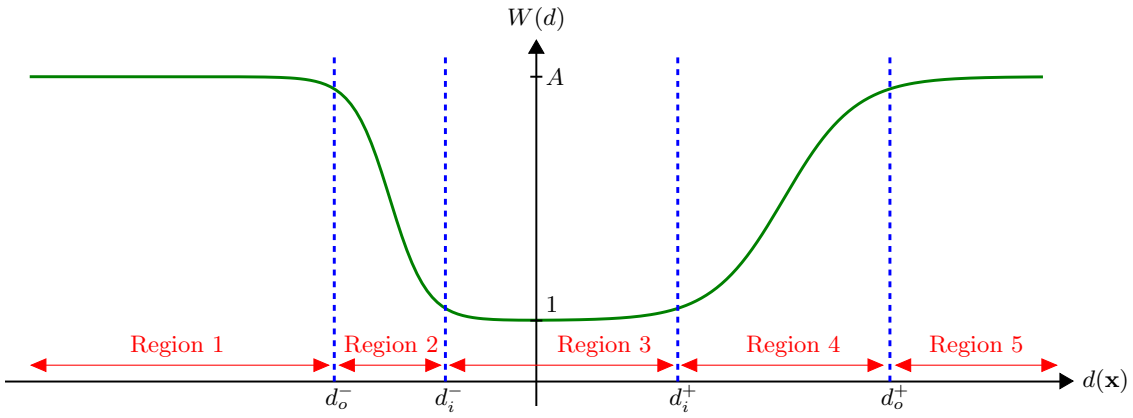


Figure 3: Qualitative plot of the signed distance function to weight function mapping.

function, $d(\mathbf{x})$. In addition to having the sign tell you which region you are in, the magnitude of the signed distance function is the shortest distance to the interface. This property makes the signed distance function an excellent choice for dynamic mesh adaptation to moving fronts. The process of going from a general level set to a signed distance function is called re-distancing or re-initialization and there are a few different algorithms for the procedure available [14, 15].

We use a piecewise-quadratic discontinuous Galerkin projection to represent the signed distance function,

$$d^c(\mathbf{x}) = \sum_{n=1}^{N} d_n^c \, b_n^c(\mathbf{x}). \tag{3.1}$$

The degrees of freedom can be computed from an L_2 projection of the distance function onto the basis functions. This is the same procedure used to compute the constant and linear degrees of freedom of the weight function in Section 2.2, except now all the degrees of freedom (including quadratic) are evaluated from the projection. As a result, the least-squares reconstruction and the WENO limiter are not used. Since the focus of this work is on the mesh relaxation method, the discontinuous Galerkin projection for the signed distance function will be computed from the exact solution. A discontinuous Galerkin based method for dynamic re-distancing on arbitrary meshes is described in Greene et al. [16].

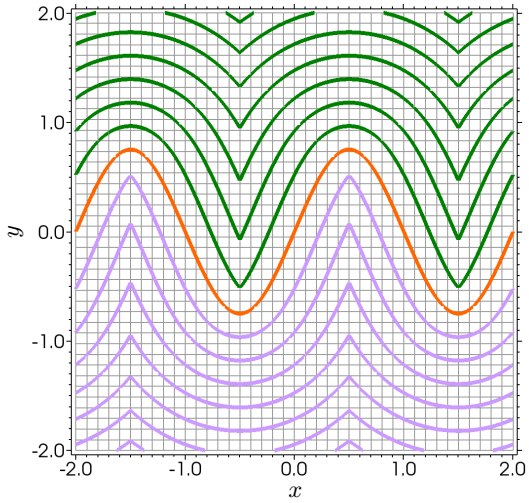
3.2 Signed distance function to weight function mapping

Since differences in the magnitude of the weight function will result in differently sized mesh cells, with smaller values generating smaller cells, the weight function should have a minimum near regions of interest. To provide extra control over the locations and size of the refinement areas, we use the "signed distance function to weight function" mapping[2] shown in Figure 3. The weight function in this figure is divided into five regions. Region 3 includes the interface and also encapsulates the area of the minimum of the weight function, which should result in the smallest mesh cells. Regions 1 and 5 contain the maximum of the weight function in the negative and positive directions of the signed distance function, respectively. This should result in the largest cells being located in these regions. Regions 2 and 4 are transitional areas, where the cell sizes will smoothly vary from the fine-mesh area in region 3 to the coarse mesh in regions 1 and 5, respectively.

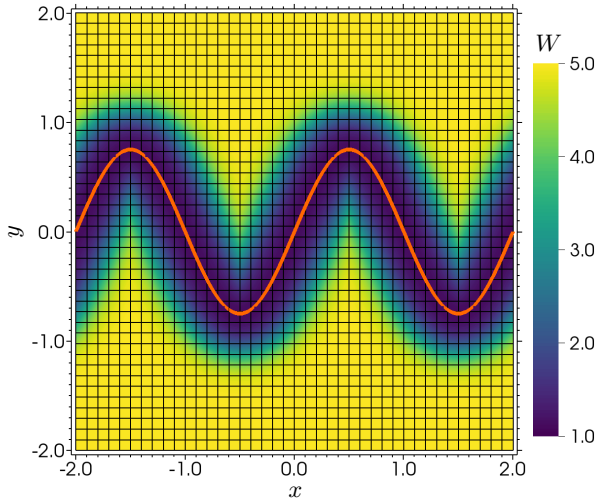
The weight function mapping from Figure 3 can be expressed as

$$W(d) = \frac{A-1}{2}\left[\tanh\left(S^+\left(d-d_m^+\right)\right) - \tanh\left(S^-\left(d-d_m^-\right)\right)\right] + A, \tag{3.2}$$

[2]Instead of prescribing the weight function W, Váchal and Maire set the inverse of it. We found it is simpler to control the distribution of the weight function when W is prescribed directly.



(a) Signed distance function. 15 evenly-spaced levels from -1.5 to 1.5. Interface —; Negative values —; Positive values —.



(b) Weight function. Interface shown as —.

Figure 4: Signed distance function and weight function for the sinusoidal interface using level set based weights. Plots shown on initial uniform mesh.

A	d_o^+	d_i^+	d_i^-	d_o^-
5.0	0.6	0.1	-0.1	-0.6

Table 1: Weight function parameters for the sinusoidal interface.

where

$$d_m^{\pm} = \frac{d_i^{\pm} + d_o^{\pm}}{2} \qquad \text{and} \qquad S^{\pm} = 2\frac{\tanh^{-1}(\zeta)}{|d_i^{\pm} - d_o^{\pm}|}. \tag{3.3}$$

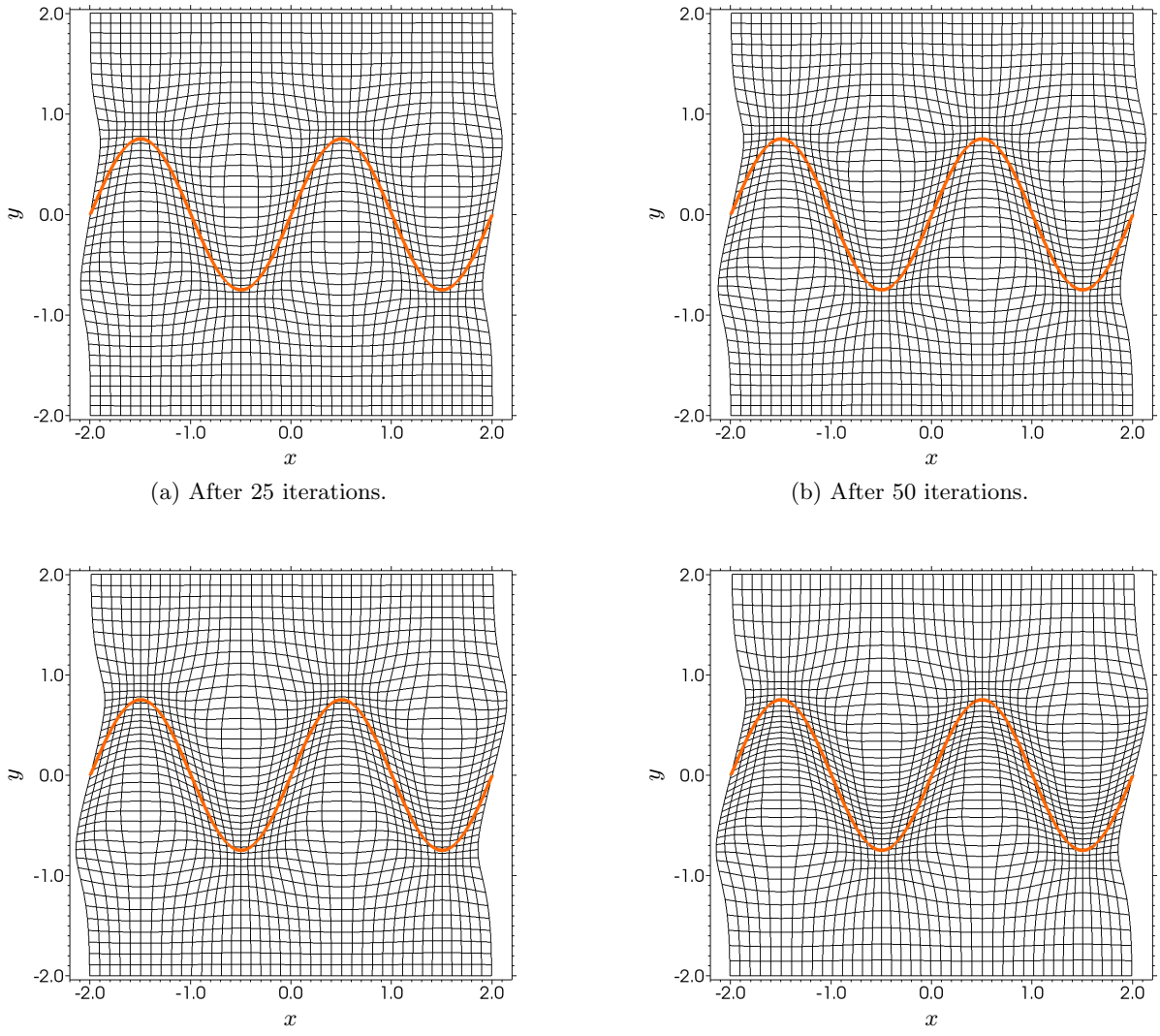
A is the weight function amplitude and it controls the amount of refinement located in region 3. ζ controls the amount the weight function has changed from A at $d_o^{\pm}$ and from 1 at $d_i^{\pm}$. A value of 0.9 was used for ζ in this work.

The mapping defined by Equation (3.2) provides far more control over the weight function distribution than what is traditional available in weighted Winslow smoothing. In Winslow smoothing, the weights are used as a scaling for the Jacobian of a cell and the value is usually defined by a material property. This results in roughly uniform cells on each side of a material interface, with the ratio of their characteristic lengths directly proportional to the ratio of their weights. The mapping used in this work provides more options than this simple binary distribution. Although we focus on refining the mesh near interfaces, the values of $d_o^{\pm}$ and $d_i^{\pm}$ can be adjusted to provide refinement at some distance away from the interface. The values can also be altered to mimic the material property distribution used in Winslow smoothing. For example, setting d_o^- and d_i^- to very large negative values will provide an approximately uniform distribution in the material defined by the negative value of the distance function. The values of d_i^+ and d_o^+ can then be adjust to provide a smooth variation of mesh resolution away from the interface. This method is used in the examples containing small and thin objects requiring a uniform mesh resolution.

3.3 Numerical examples

3.3.1 Sinusoidal interface

The first example for the level set based CNR is a sinusoidal interface. Figure 4a shows a contour plot of the signed distance function for the interface on the initial 41×40 mesh. This geometry was selected to



(a) After 25 iterations.

(b) After 50 iterations.

(c) After 100 iterations.

(d) After 500 iterations.

Figure 5: Mesh generated for the sinusoidal interface using level set based weights. Mesh shown after various iteration steps. Started from a uniform mesh with 41×40 cells. Interface shown as —.

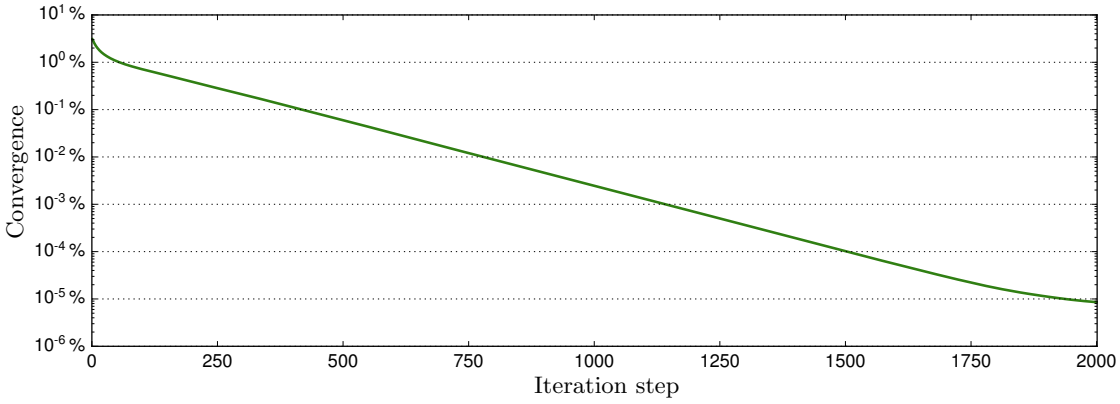


Figure 6: Convergence for the sinusoidal interface using level set based weights.

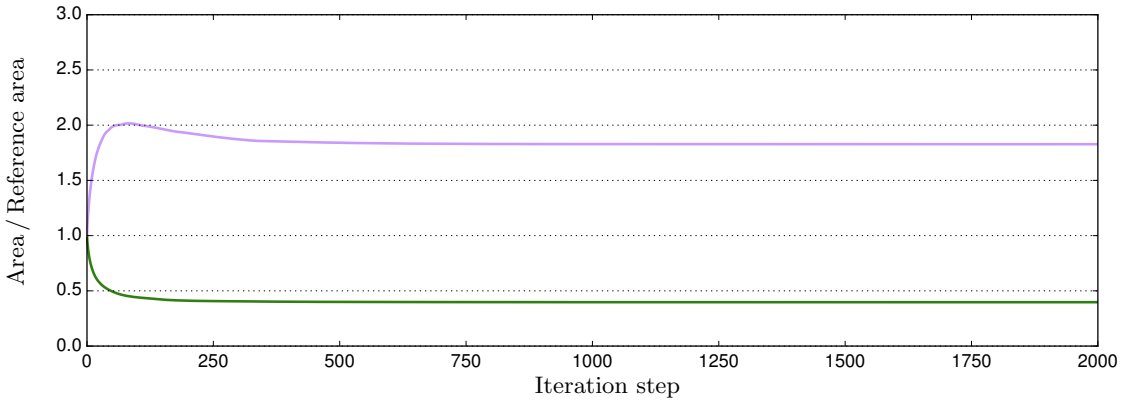


Figure 7: Minimum and maximum cell area for the sinusoidal interface using level set based weights. Minimum area —; Maximum area —;

A	d_o^+	d_i^+	d_i^-	d_o^-
5.0	0.6	0.1	-1.0	-2.0

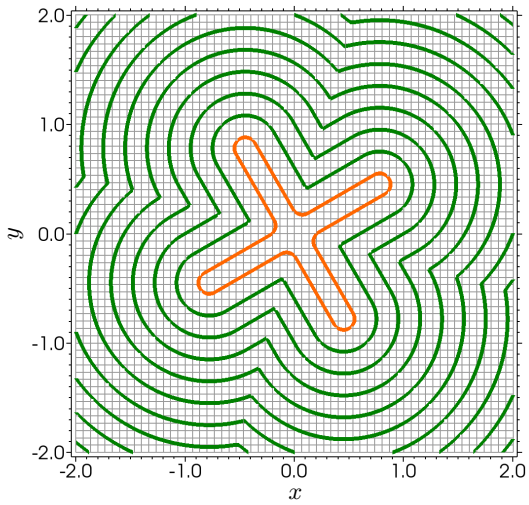
Table 2: Weight function parameters for the cruciform interface.

approximate the linear growth stage of the Rayleigh-Taylor instability. This mesh uses periodic boundary conditions in the x direction. For this example, we want to increase the mesh resolution symmetrically around the interface. This can be accomplished by using the weight function shown in Figure 4b. The parameters for the weight function are given in Table 1. The resulting mesh after various numbers of iterations is shown in Figure 5. This mesh possesses the desired refinement along the interface, with no difficulty following the geometry. A history of the convergence measured by Equation (2.11) is shown in Figure 6 for 2000 iterations. The method converged to approximately 0.5% after about 150 iterations. This can also be seen from the visually indistinguishable change after iteration 100 in Figure 5. Figure 7 shows a history of the minimum and maximum cell area of the mesh. The plot confirms the convergence seen in Figure 6.

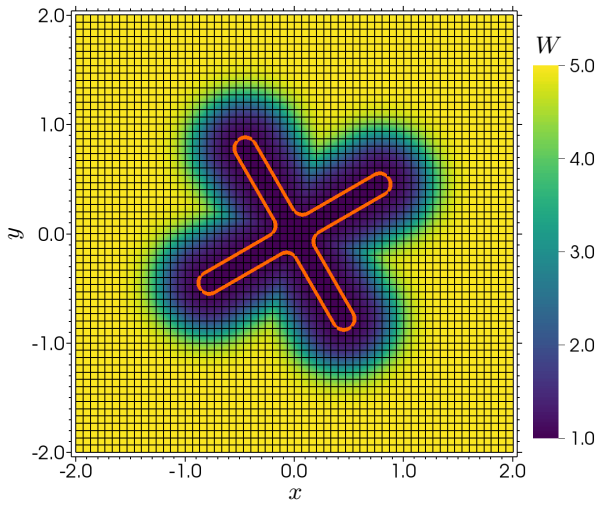
3.3.2 Cruciform interface

The second example being presented is a cruciform geometry. The signed distance function for the geometry is shown in Figure 8a on the initial 60×60 uniform mesh. The length of each arm of the cruciform is 1 and the thickness is 0.2. The corners of the cruciform are rounded with a radius of 0.1. The cruciform is rotated 30° from the horizontal. The cruciform geometry was selected to test the mesh relaxation method on a complex geometry with thin regions requiring increased resolution. In addition, this geometry has a number of discontinuities in the slope of the signed distance function. For the positive values, they are located between the arms of the cruciform and appear to originate at the corners. Although not evident with the current contours, there are also discontinuities in the negative values along the center of each arm. For this case, it is desirable to generate a high resolution region with approximately uniform cells within the cruciform and then transition to a coarser resolution away from the interface. The weight function shown in Figure 8b was designed to accomplish this. The weight function parameters are given in Table 2.

Figure 9 shows the mesh generated from the weight function shown in Figure 8b after different numbers of relaxation iterations. The generated mesh has clearly conformed to the cruciform geometry and had no issues dealing with the discontinuities in the signed distance function. In addition, the mesh possesses the desired high resolution inside the cruciform and smoothly transitions to a less dense resolution away from the cruciform. Initially, the arms of the cruciform are highly under-resolved, with only about two cells covering the width of the arms. After 500 iterations, the resolution has doubled to approximately four cells. Figure 10 shows the convergence measure for 2000 relaxation iterations. Figure 11 shows the history of the minimum and maximum cell area. The plots show a slightly slower convergence compared to the sinusoidal interface,



(a) Signed distance function. 10 evenly-spaced levels from 0.0 to 1.8. Interface —; Positive values —.



(b) Weight function. Interface shown as —.

Figure 8: Signed distance function and weight function for the cruciform interface using level set based weights. Plots shown on initial uniform mesh.

but this is expected due to the increased number of mesh cells.

3.3.3 Two level sets example

In multi-material simulations, different material interfaces are usually tracked using separate signed distance functions and thus have different weight functions associated with them. Since they represent different materials, their topologies may vary greatly in size and thus require different levels of interfacial mesh resolution. This can be addressed by setting different amplitudes for the weight function computed from each signed distance function. Simply using Equation 3.2 with different values for the amplitude, A, is inadvisable, since the different functions will have the same minimum and different maximums. Instead, we rescale the individual weight functions such that they all have the same maximum, while still maintain the same ratio between their maximum and minimum values.

Let A_ℓ be the prescribed amplitude for the weight function computed from the ℓth signed distance function. The rescaled weight function mapping for the ℓth distance function, $W_\ell^*(d)$, will take the form

$$W_\ell^*(d) = \frac{A_\ell^* - B_\ell^*}{2}\left[\tanh\left(S^+\left(d - d_m^+\right)\right) - \tanh\left(S^-\left(d - d_m^-\right)\right)\right] + A_\ell^*, \tag{3.4}$$

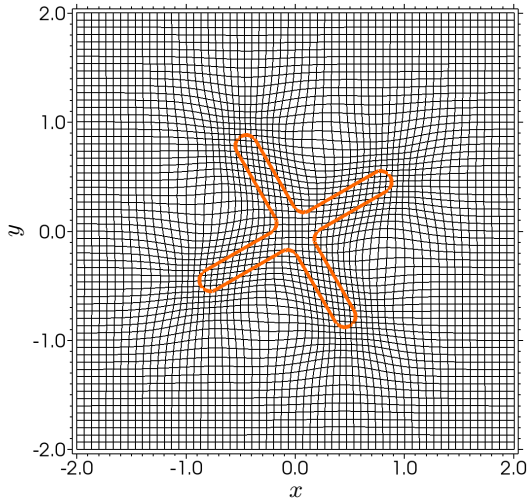
where A_ℓ^* and B_ℓ^* are the rescaled maximum and minimum of the weight function, respectively. To ensure that all the weight functions have the same maximum,

$$A_\ell^* = \max_{\ell \in \mathcal{L}} \left(A_\ell\right), \tag{3.5}$$

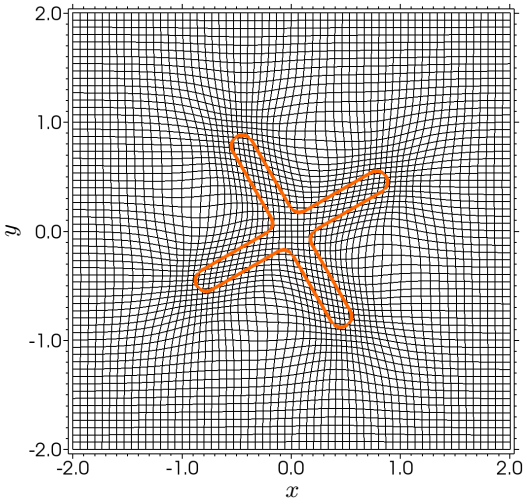
where $\mathcal{L}$ is the set of all the signed distance functions. B_ℓ^* is then set to maintain the ratio between the maximum and minimum values of the weight function,

$$B_\ell^* = \frac{A_\ell^*}{A_\ell}. \tag{3.6}$$

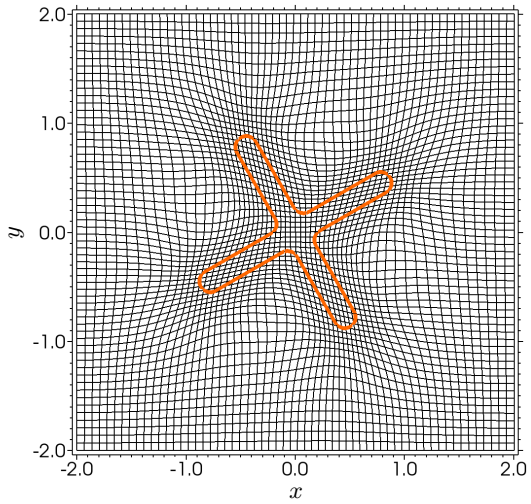
By using these equations, a weight function for each distance function can be calculated. However, the formulation of the condition number in Equation (2.1) has only a single value for the weight function, so the weights from the distance functions for each material must be combined in some manner. This



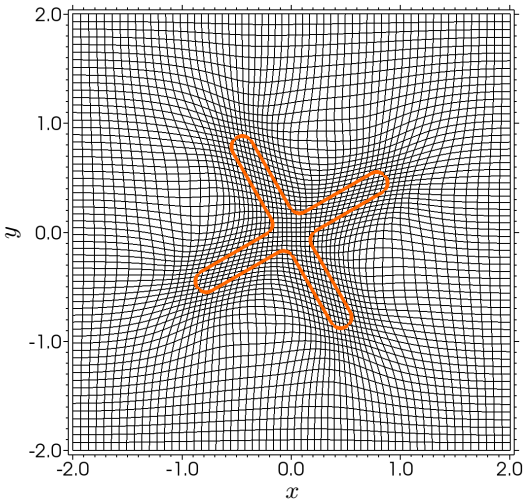
(a) After 50 iterations.



(b) After 100 iterations.



(c) After 300 iterations.



(d) After 500 iterations.

Figure 9: Mesh generated for the cruciform interface using level set based weights. Mesh shown after various iteration steps. Started from a uniform mesh with 60×60 cells. Interface shown as —.

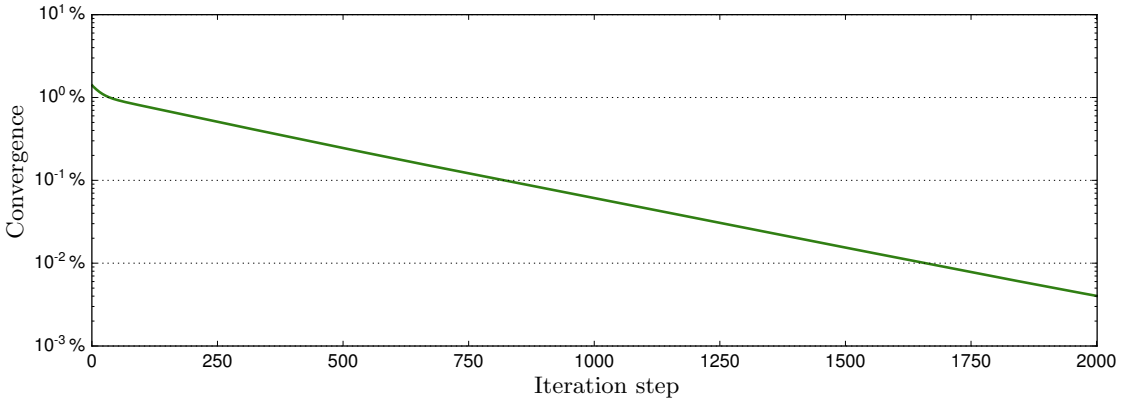


Figure 10: Convergence for the cruciform interface using level set based weights.

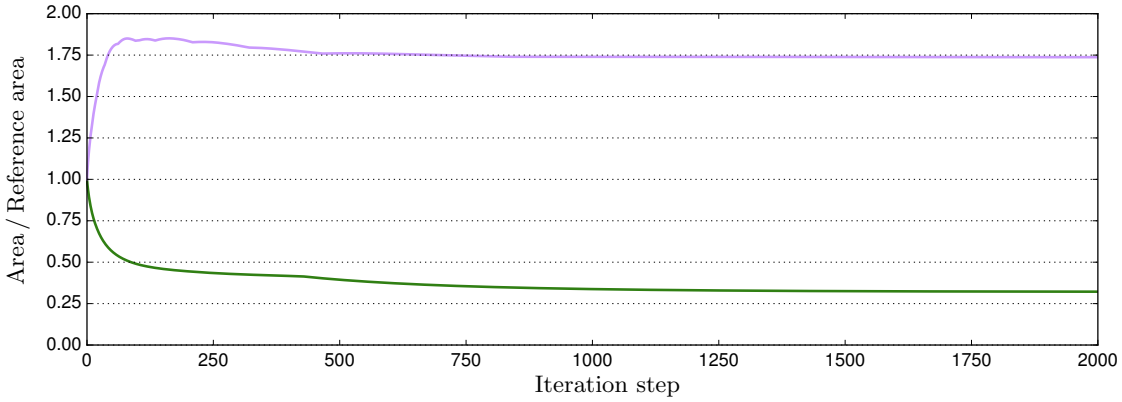
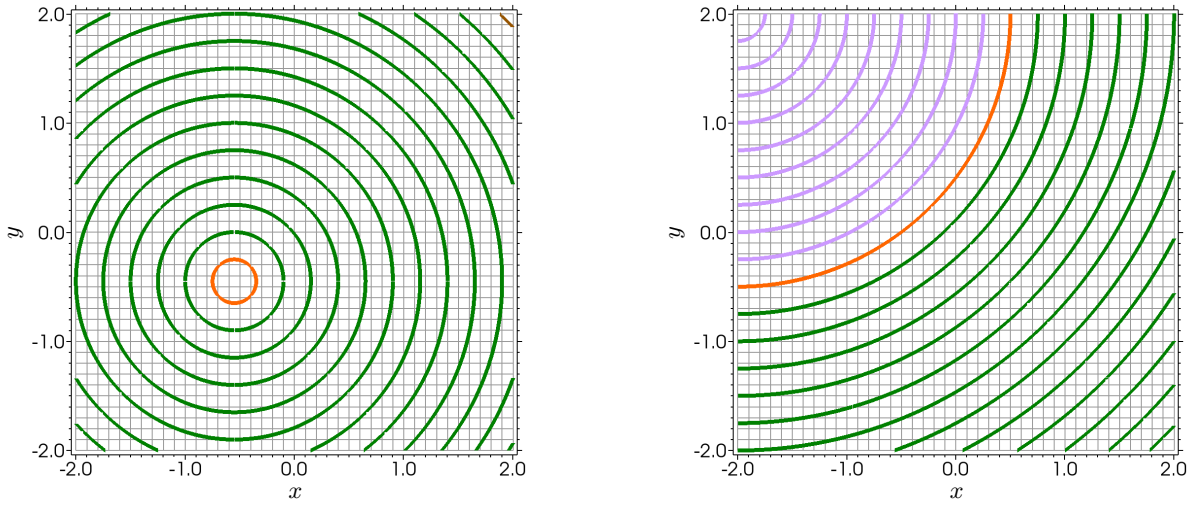


Figure 11: Minimum and maximum cell area for the cruciform interface using level set based weights. Minimum area —; Maximum area —;

	Center	Radius	A	d_o^+	d_i^+	d_i^-	d_o^-
Small Circle	(-0.55, -0.45)	0.2	5.0	0.6	0.1	-1.0	-2.0
Large Circle	(-2.0, 2.0)	2.5	2.5	0.6	0.1	-0.1	-0.6

Table 3: Geometric and weight function parameters for the two level sets example.



(a) Signed distance function for the small circle.

(b) Signed distance function for the large circle.

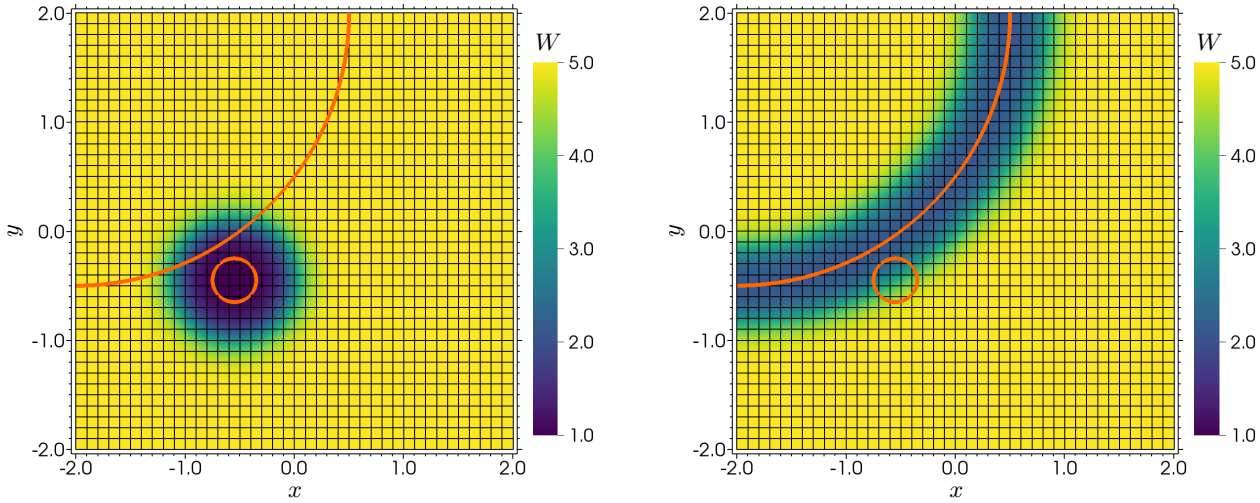
Figure 12: Signed distance functions for the two level sets example. Plots shown on initial uniform mesh. 24 evenly-spaced levels from -2.5 to 3.25. Interface —; Negative values —; Positive values —.

can be addressed by simply using the minimum value of the weight computed from each distance function when computing the discontinuous Galerkin projection of the combined weight function. This results in Equation (A.1) being replaced with

$$\int_{\Omega_c} \min_{\ell \in \mathcal{L}} \left(W_\ell^*(\mathbf{x})\right)\, b_m^c(\mathbf{x})\, d\Omega = \sum_{n=1}^{N} W_n^c \int_{\Omega_c} b_n^c(\mathbf{x})\, b_m^c(\mathbf{x})\, d\Omega. \tag{3.7}$$

As an example, we generated a mesh for two circular interfaces whose radii differ by an order magnitude. Figure 12 shows the signed distance function for each circle on the initial 40×40 mesh. This can represent a simulation with three different materials. Given the large difference in radii, the smaller circle will require a significantly higher resolution to accurately describe its geometry. This is accomplished by using the weight function from Figure 13a for the smaller circle. For the larger circle, much less resolution is needed compared to the small circle, which is accomplished by the weight function shown in Figure 13b. Table 3 has the geometric parameters and weight function parameters used for both circles. The weight function resulting from combing the two individual functions using Equation (3.7) is shown in Figure 13c. By using the minimum value of the two functions, the combined function will ensure the smaller circle is able to maintain sufficient resolution near the larger circle.

The resulting mesh at various iteration steps is shown in Figure 14. Just as desired, the mesh possess a small amount of clustering along the large circle's interface and a large amount of clustering around the small circle. The initial mesh has approximately four mesh cells across the diameter of the small circle. After 500 iterations, the resolution has increased to approximately seven cells across the diameter. Figure 15 shows the convergence measure for 2000 iterations. Figure 16 shows the minimum and maximum cell area for the same iterations. The figures show that the mesh converges at a similar rate as the previous sinusoidal example.

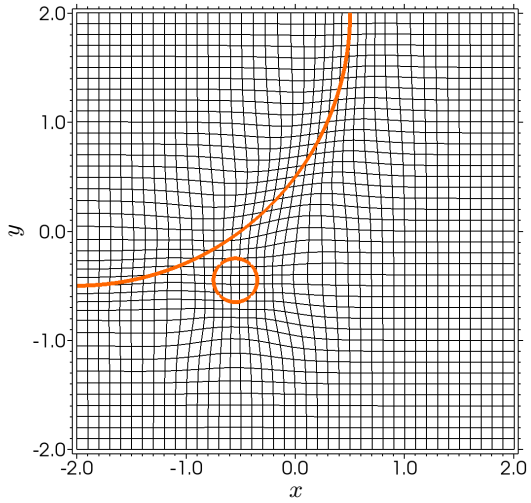


(a) Weight function for the small circle.

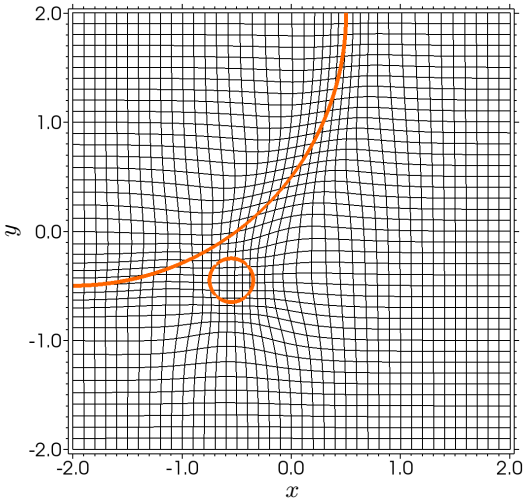
(b) Weight function for the large circle.

(c) Combined weight function.

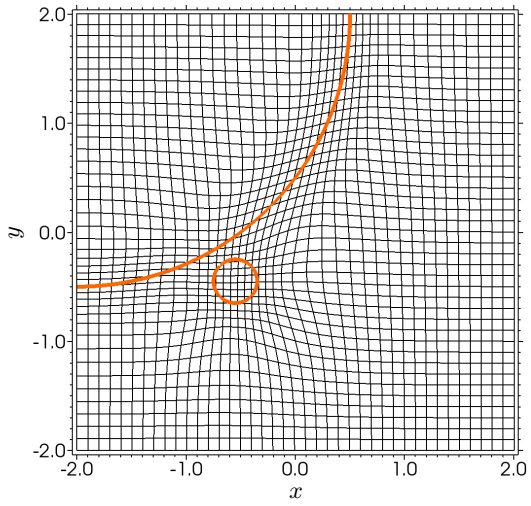
Figure 13: Weight functions for the two level sets example using level set based weights. Plots shown on initial uniform mesh. Interfaces are shown as —.



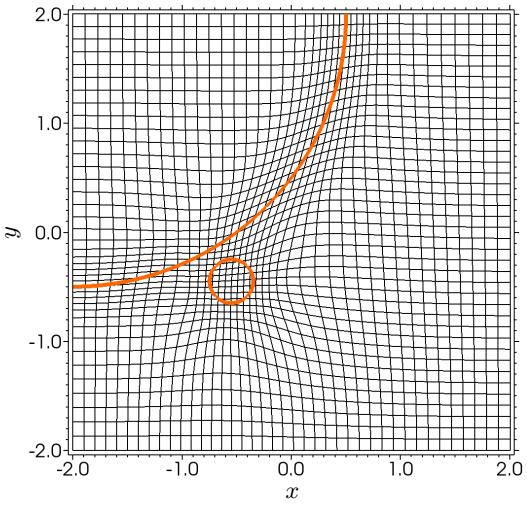
(a) After 25 iterations.



(b) After 50 iterations.



(c) After 100 iterations.



(d) After 500 iterations.

Figure 14: Mesh generated for the two level sets example using level set based weights. Mesh shown after various iteration steps. Started from a uniform mesh with 40×40 cells. Interfaces are shown as —.

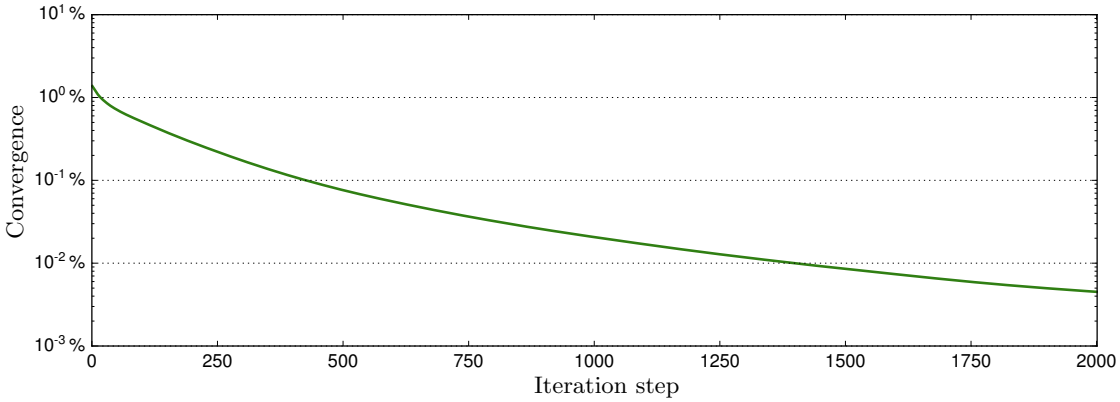


Figure 15: Convergence for the two level sets example using level set based weights.

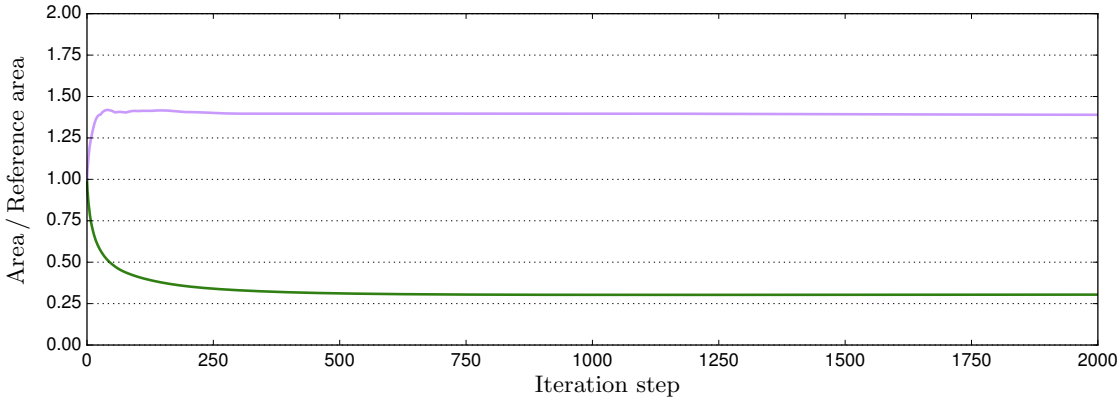


Figure 16: Minimum and maximum cell area for the two level sets example using level set based weights. Minimum area —; Maximum area —;

	Center	Radius	A	d_o^+	d_i^+	d_i^-	d_o^-
Circle	(0.0, 0.0)	1.25	2.0	0.6	0.1	-1.0	-0.5
Right Semicircle	(0.33, 0.11)	0.5	4.0	0.6	0.0	-1.0	-2.0
Left Semicircle	(-0.51, -0.26)	0.3	9.0	0.5	0.0	-1.0	-2.0

Table 4: Geometric and weight function parameters for the three level sets example.

3.3.4 Three level sets example

The final example for the level set based CNR is an example involving three level sets, which can represent four different materials. In this example, a mesh will be generated for a circular interface with two semi-circle interfaces within it. Figure 17 depicts the signed distance function for each interface on the initial 50×50 uniform mesh. Similar to the last example, since the interfaces have different scales, different amounts of clustering will be needed. Due to its large size, it is assumed that only a small amount of refinement is needed along the circle. Given the smaller size of the semi-circles, a larger amount of refinement will used for them. To achieve this, different weight functions were generated for each geometry and are shown in Figure 17. The figure also includes the combined weight function that will be used for the mesh relaxation procedure. Table 4 provides the geometric and weight function parameters applied in this example.

Figure 19 shows the mesh after various iteration steps. As in all the previous cases, the mesh is able to conform to the prescribed geometries and achieve the desired mesh resolution distribution. The convergence for each iteration is shown in Figure 20, and the minimum and maximum cell area are presented in Figure 21. As one can see from these figures, the mesh has nearly converged after approximately 500 iterations.

4. Index function based weight function

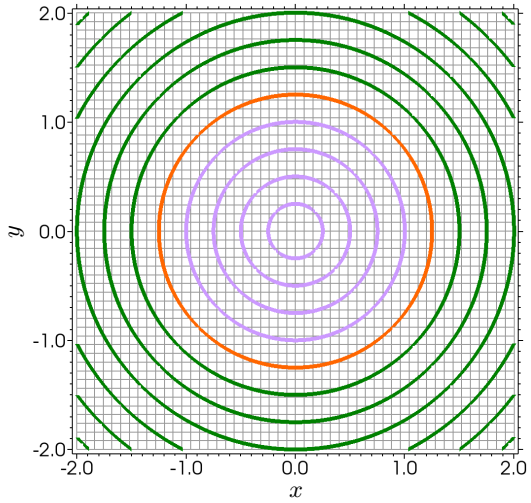
The level set based weights from the previous section were shown to achieve the desired result of clustering mesh cells near interfaces. However, many existing ALE codes do not use a level set method for interface tracking and it may not always be feasible to add a level set method to the code. Without a level set method, these codes usually utilize an index function to track the location of materials. An index function will have a single value for all cells on one side of the interface and a different value for all cells on the other side of the interface. The value for cells that contain a portion of the interface will be a volume fraction of the two extremes. The index function would usually be stored as a discrete cell-centered quantity. An example of this is the volume fraction from the volume of fluid method [17]. For these cases, we have developed a method to generate similar meshes using an index function instead of a level set.

4.1 Low-order signed distance function

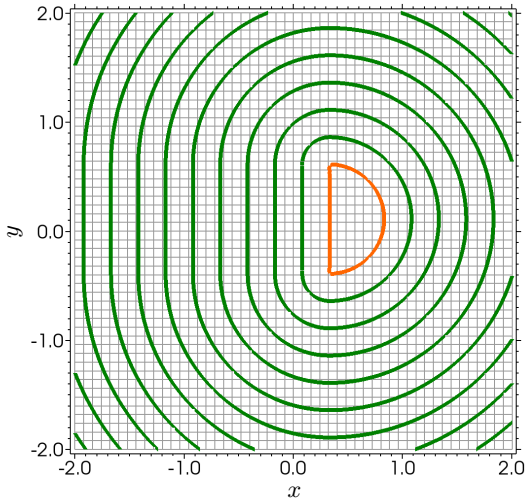
The idea behind the index function based weights is to generate a low-order signed distance function (LOSDF) from the index function and then use the LOSDF in Equation (3.2) to compute the weight function. The LOSDF is generated by first finding an approximation of the value of the signed distance function at the centroid of each cell. This value is then used as the first degree of freedom for a piecewise-quadratic discontinuous Galerkin projection. The remaining degrees of freedom are generated from a least-squares reconstruction using the face and vertex neighbors of the cell. This projection is often abbreviated rDG(P_0P_2) in the literature.

For the LOSDF, we assume the index function is $+1$ for cells totally on one side of the interface, -1 for cells totally on the other side of the interface, and a volume fraction for interface cells. Let V^c represent the index function in cell c. The first step of the method is to compute the first degree of freedom in each of the interface cells. The value can be approximated as

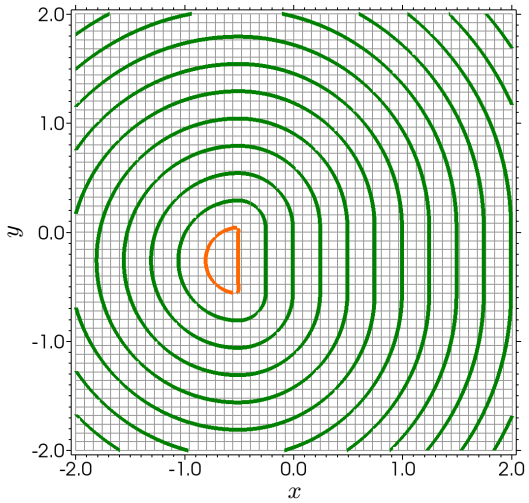
$$d_1^c = \left(\frac{V^c}{2}\right)\left(\frac{\Delta x + \Delta y}{2}\right). \tag{4.1}$$



(a) Signed distance function for the circle.

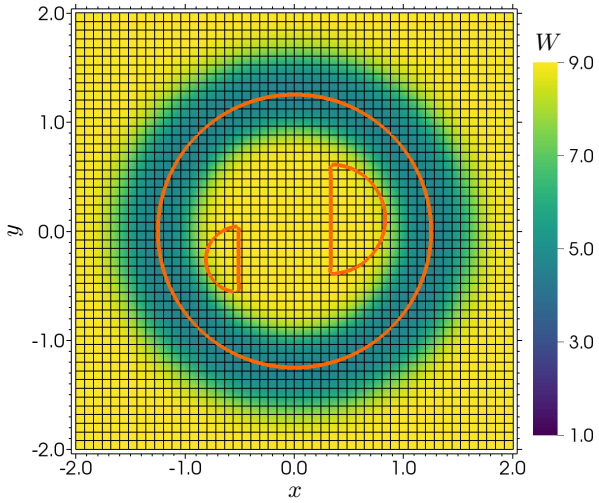


(b) Signed distance function for the right semicircle.

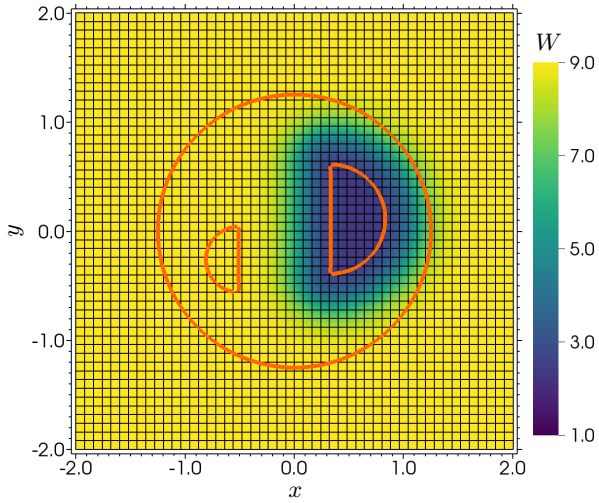


(c) Signed distance function for the left semicircle.

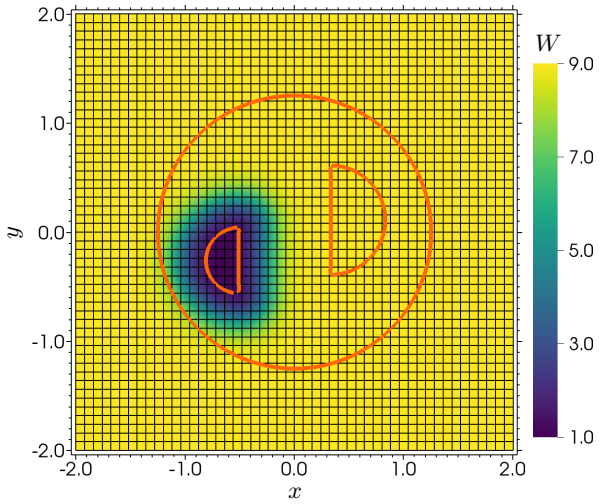
Figure 17: Signed distance functions for the three level sets example. Plots shown on initial uniform mesh. 17 evenly-spaced levels from -1.0 to 3.0. Interface —; Negative values —; Positive values —.



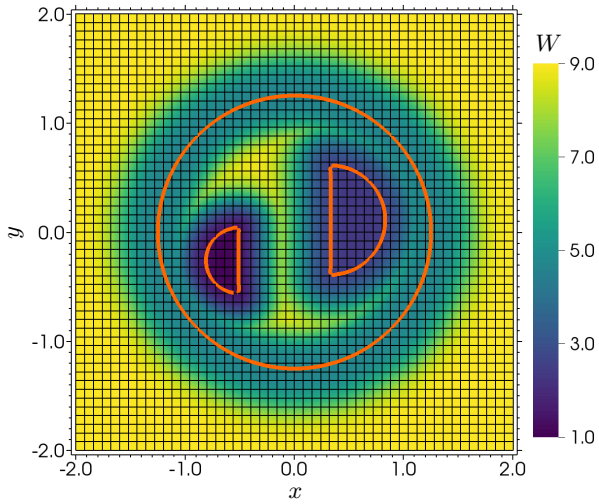
(a) Weight function for the circle.



(b) Weight function for the right semicircle.

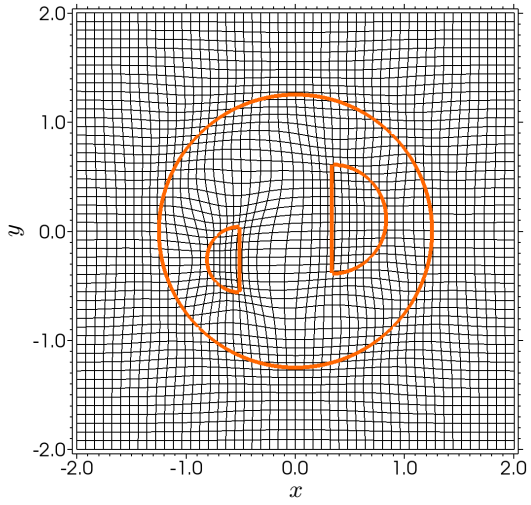


(c) Weight function for the left semicircle.

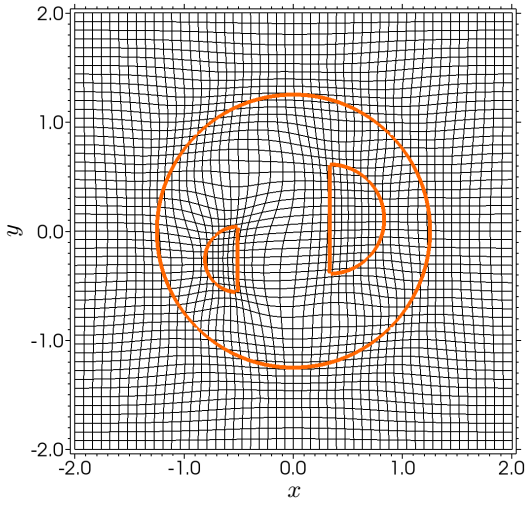


(d) Combined weight function.

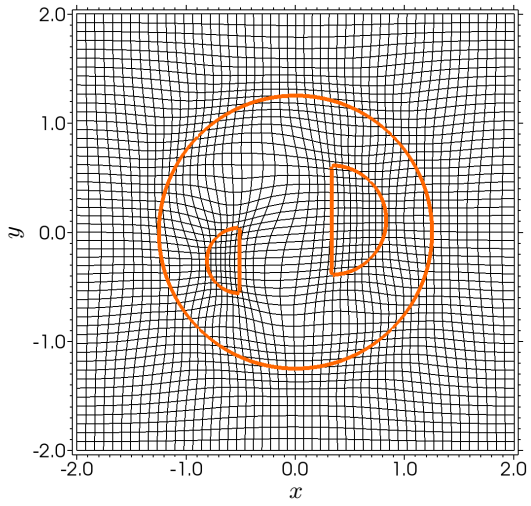
Figure 18: Weight functions for the three level sets example using level set based weights. Plots shown on initial uniform mesh. Interfaces are shown as —.



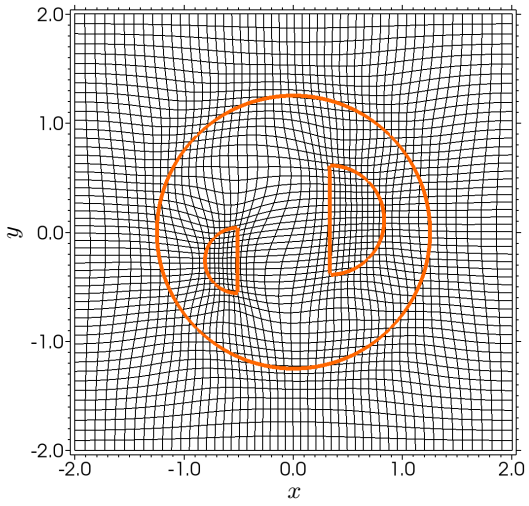
(a) After 25 iterations.



(b) After 50 iterations.



(c) After 100 iterations.



(d) After 500 iterations.

Figure 19: Mesh generated for the three level sets example using level set based weights. Mesh shown after various iteration steps. Started from a uniform mesh with 50×50 cells. Interfaces are shown as —.

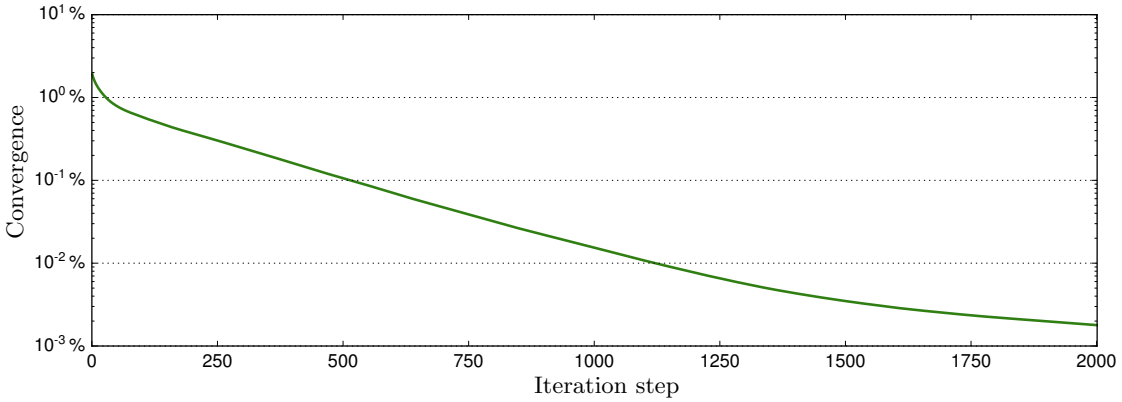


Figure 20: Convergence for the three level sets example using level set based weights.

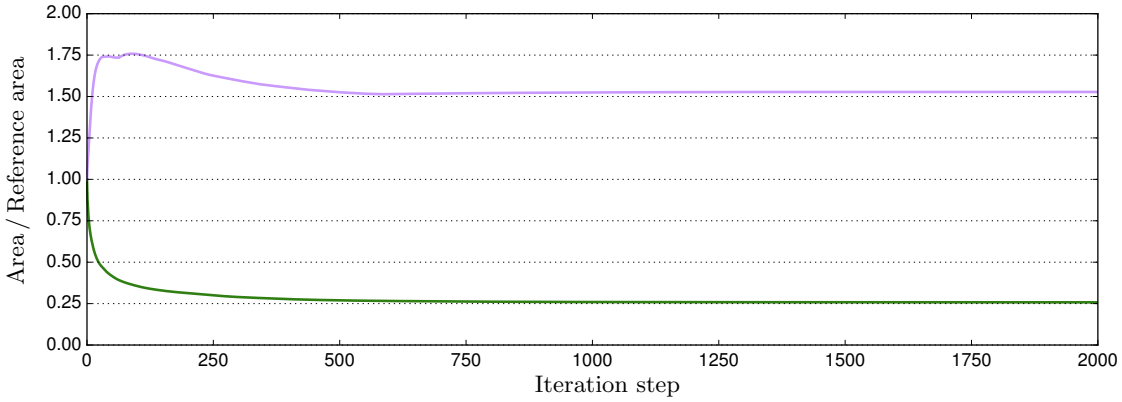
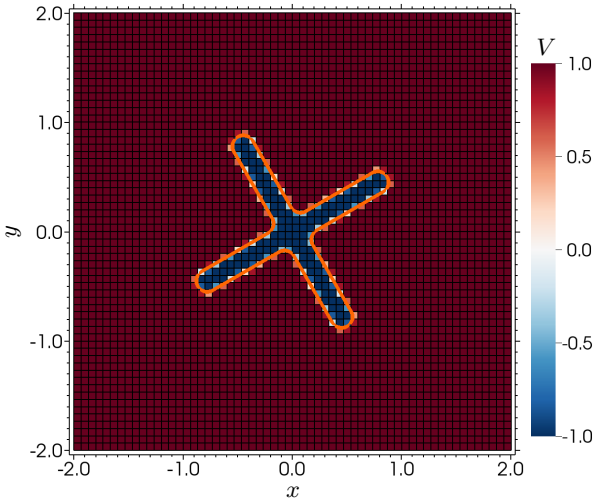
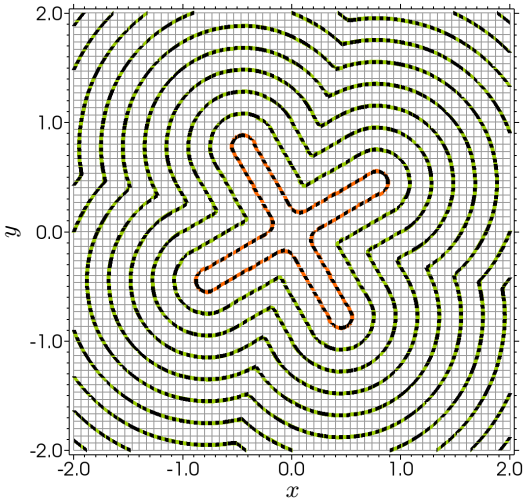


Figure 21: Minimum and maximum cell area for the three level sets example using level set based weights. Minimum area —; Maximum area —;



(a) Index function. Interface shown as —.



(b) Low-order signed distance function. 10 evenly-spaced levels from 0.0 to 1.8. Interface —; Positive values —; Exact distance function - - -.

Figure 22: Index function and low-order signed distance function for the cruciform interface.

With all the interface cells known, the value in the remaining cells can be computed from the expression

$$d_1^c = \text{sign}(V^c) \min_{i \in I} \Big(\big| d_1^i + \text{sign}(V^c) D_{c,i} \big| \Big), \tag{4.2}$$

where $D_{c,i}$ is the distance between the centroid of cell c and the centroid of cell i, and I is the set of all the interface cells. At this point, the first degree of freedom has been computed in every cell and the reconstruction can be applied to compute the remaining degrees of freedom. The details of the rDG(P_0P_2) reconstruction are provided in Appendix B. With the LOSDF projection known, it can be used to compute the weight function in the same manner as the exact signed distance function from Section 3.

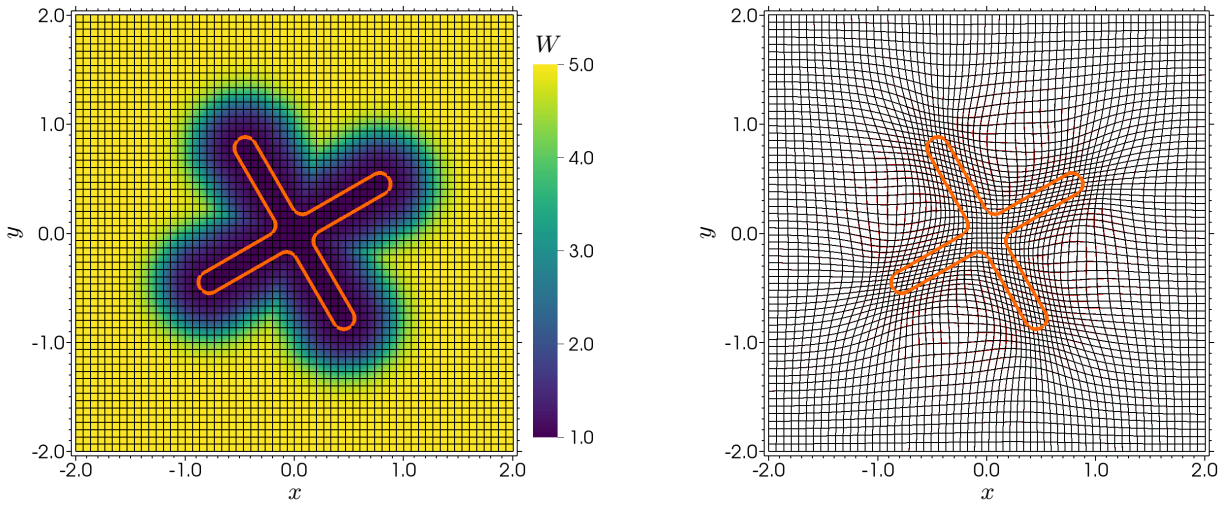
4.2 Numerical examples

4.2.1 Cruciform interface

The first example is the index function version of the cruciform from Section 3.3.2. Figure 22a shows the index function used to define the geometry. The LOSDF generated from the index function is shown in Figure 22b. For comparison, the exact distance function is also included in the plot. The LOSDF does a surprisingly good job of estimating the exact signed distance function, especially considering the initial index function values it was generated from and the overall simplicity of the algorithm.

To permit comparison with the level set based weights, we use the same weight function parameters. Figure 23a shows the weight function computed from the LOSDF and the resulting mesh is shown in Figure 23b. Also shown in the figure is the mesh generated using the level set based weights. The meshes generated in both cases are comparable, with only slight differences barely seen by the naked eye.

Figure 24 shows the convergence history for this example. The values for the level set based version is also included for comparison. For the first half of the mesh generation, the results from both methods are very similar. During the second half, the index function version begins to display some oscillations, but on the average, the magnitude of the mesh motion continues to decay at a rate similar to the level set version. Figure 25 shows the history for the minimum and maximum cell area, which are also very similar to the level set based approach.



(a) Weight function.

(b) Mesh after 300 iterations. Index function based weights —; Level set based weights —.

Figure 23: Weight function and mesh for the cruciform interface using index function based weights. Interface shown as —.

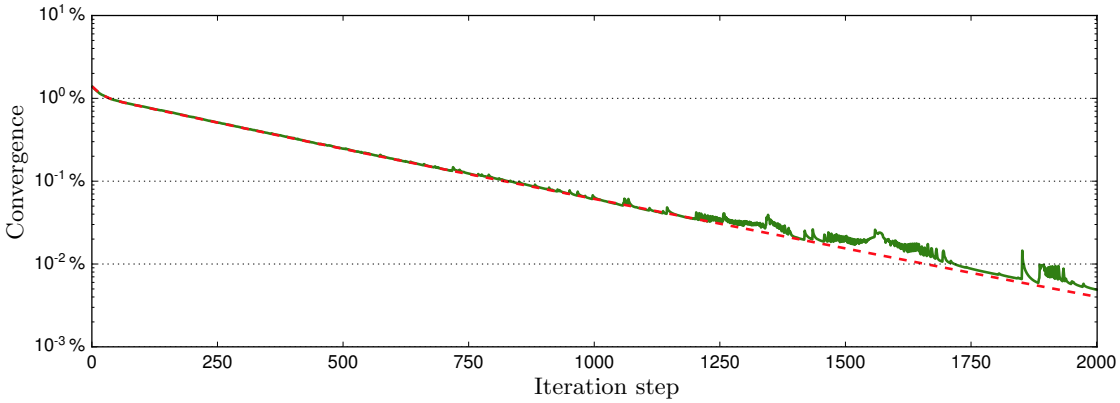


Figure 24: Convergence for the cruciform interface using index function based weights. Index function based weights —; Level set based weights - - -.

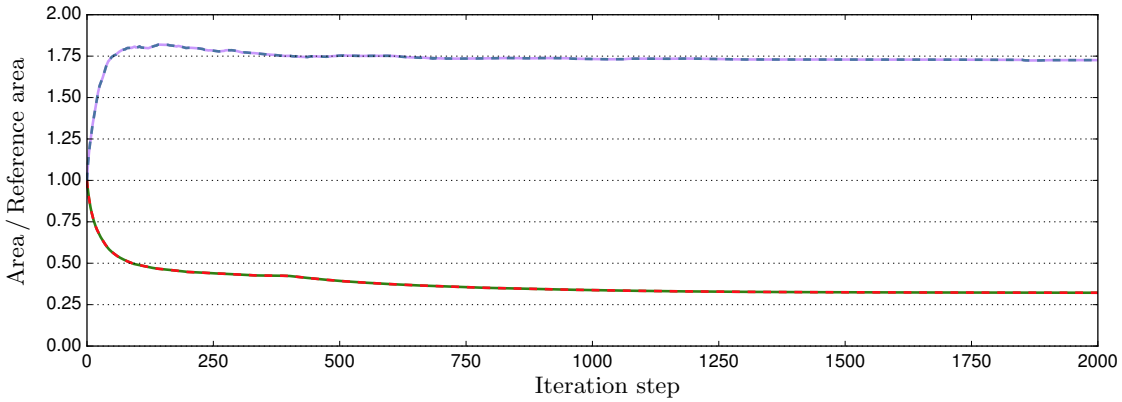


Figure 25: Minimum and maximum cell area for the cruciform interface using index function based weights. Index function based minimum area —; Index function based maximum area —; Level set based minimum area - -; Level set based maximum area - -.

	Center	Radius	A	d_o^+	d_i^+	d_i^-	d_o^-
Lower Circle	(-0.53, -1.00)	0.75	5.0	0.6	0.1	0.0	-0.4
Upper Right Circle	(1.0, 1.07)	0.5	5.0	0.6	0.1	0.0	-0.4
Upper Left Circle	(-1.13, 0.89)	0.5	5.0	0.6	0.1	0.0	-0.4

Table 5: Geometric and weight function parameters for the three circles interface.

4.2.2 Three circles interface

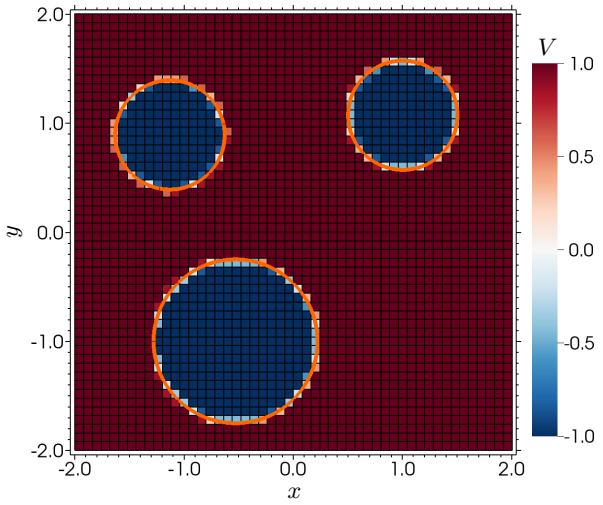
The next example is an interface defined by three circles. The index function used in this case is shown in Figure 26a, on the initial 50×50 uniform mesh. Table 5 defines the geometric parameters for the circles. Although there are three circles, they are all defined by a single index function, which will result in a single LOSDF and is shown in Figure 26b. Similar to the cruciform, the low-order function does a good job of estimating the exact signed distance function. The differences between the results are most noticeable at the discontinuities where the distance functions from each individual circle converge.

Figure 27a shows the weight function computed from the LOSDF. Since only a single LOSDF is used, only one weight function is generated. The parameters for the weight function (Table 5) were selected to cluster the mesh cells along the outer edge of each interface. The final generated mesh is depicted in Figure 27b. The differences between the meshes are more pronounced in this case than in the last test case, with the largest discrepancies seen near the distance function discontinuities between the circles. Even thought the meshes are slightly different, neither is clearly superior to the other.

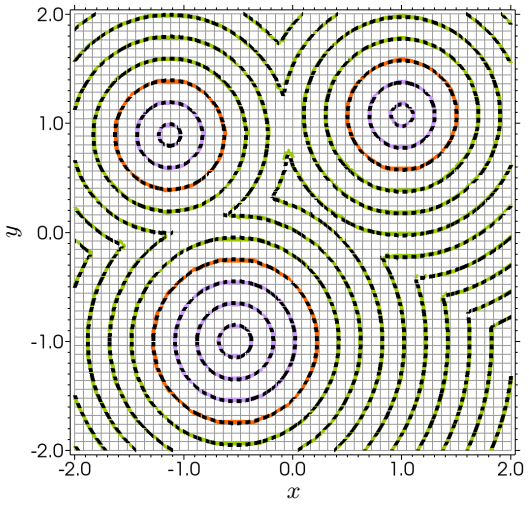
Figure 28 depicts the convergence history. During the initial iterations, the convergence follows the results from the level set based mesh. However, after about 250 iterations, the values diverge slightly. While the mesh motion for the level set based mesh continues to drop, the current simulation appears to level off at around 0.11%. This suggests there are some oscillations in the position of the vertices. However, these oscillations are very small in magnitude. The minimum and maximum cell area histories are shown in Figure 29. The values for the index function weights match those for the level set based approach.

4.2.3 Two level sets example

The last example for the index function based weights is the two level sets example from Section 3.3.3. As in the level set based approach, the two interfaces are stored separately. Figures 30a and 30c show the index function values for the large and small circles, respectively. The LOSDFs generated from the index functions for each circle are shown in Figures 30b and 30d. Figure 31a shows the weight function computed from the LOSDF and the resulting mesh is shown in Figure 31b. The generated mesh is nearly identical to the mesh

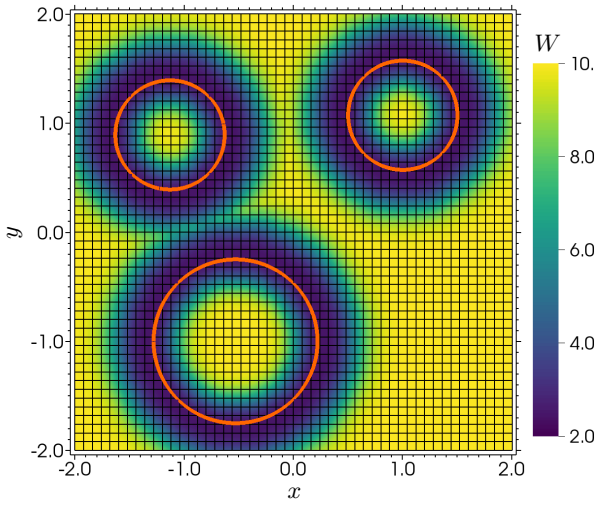


(a) Index function. Interface shown as —.

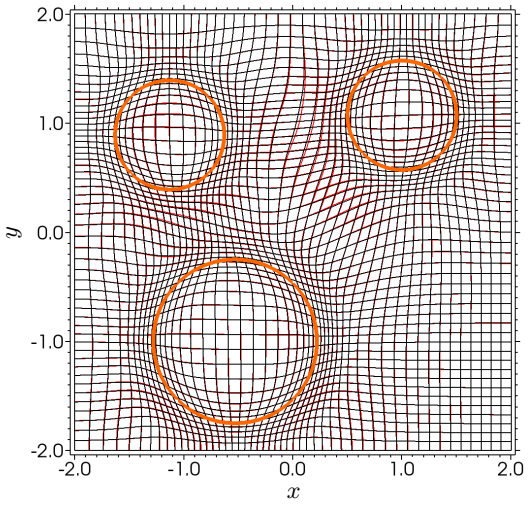


(b) Low-order signed distance function. 13 evenly-space levels from -0.6 to 1.8. Interface —; Negative values —; Positive values —; Exact distance function - -.

Figure 26: Index function and low-order signed distance function for the three circles interface.



(a) Weight function.



(b) Mesh after 200 iterations. Index function based weights —; Level set based weights —.

Figure 27: Weight function and mesh for he three circles interface using index function based weights. Interface shown as —.

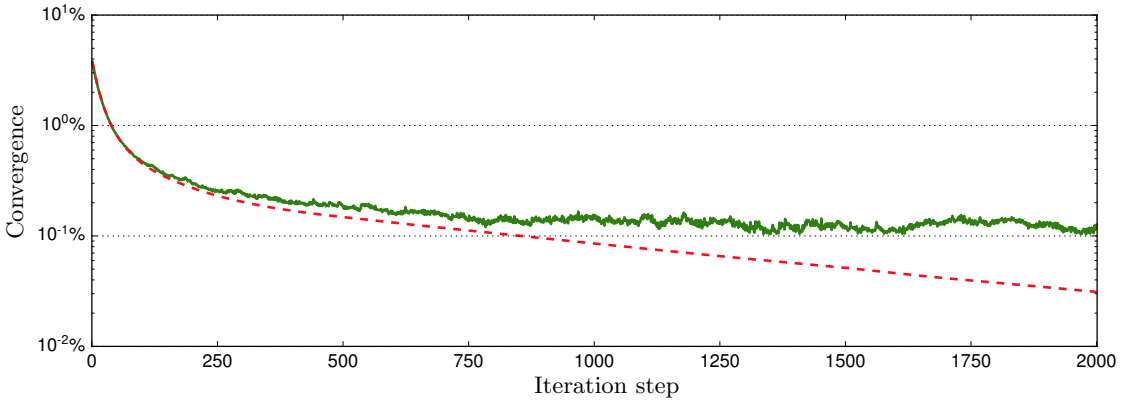


Figure 28: Convergence for the three circles interface using index function based weights. Index function based weights —; Level set based weights - -.

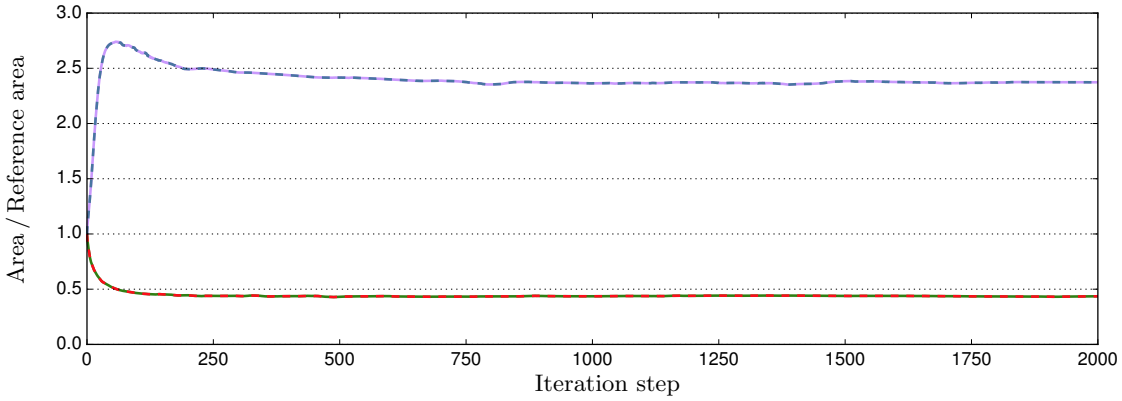
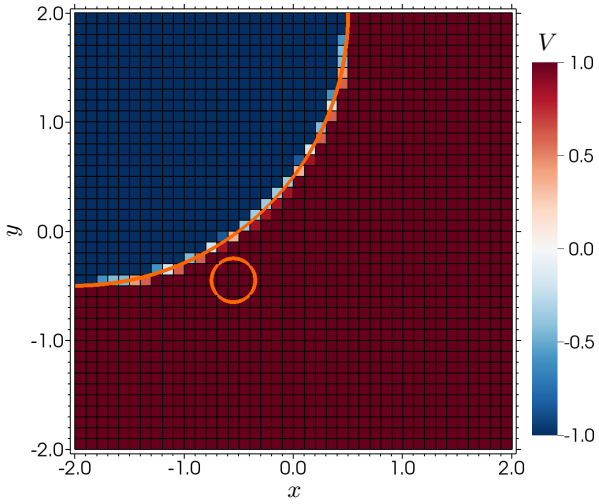
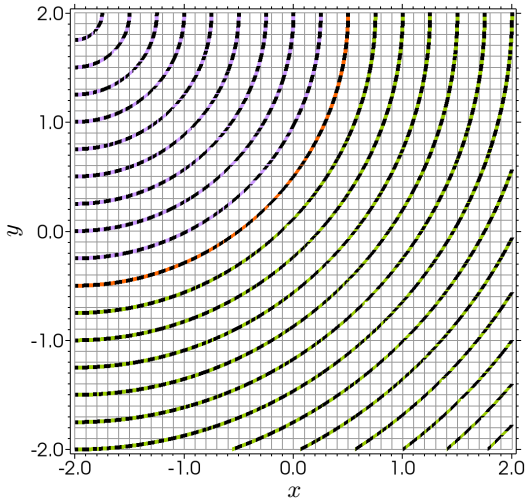


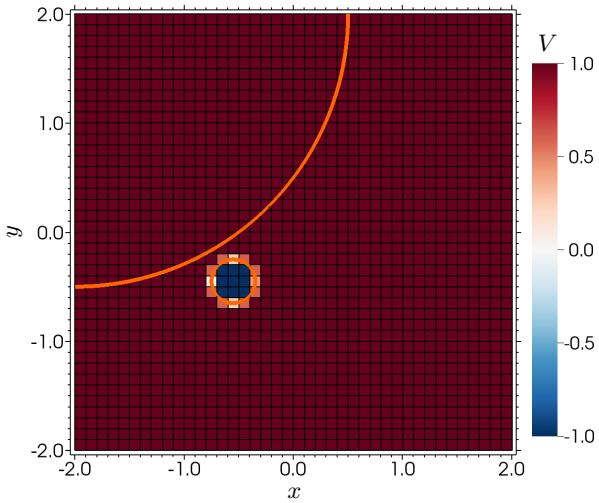
Figure 29: Minimum and maximum cell area for the three circles interface using index function based weights. Index function based minimum area —; Index function based maximum area —; Level set based minimum area - -; Level set based maximum area - -.



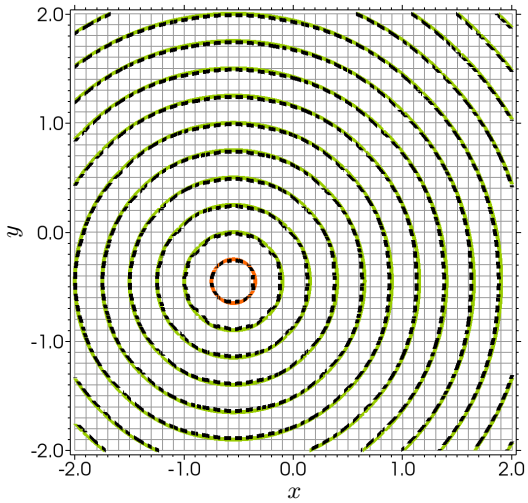
(a) Index function used to define the large circle geometry. Interfaces shown as —.



(b) Low-order signed distance function for the large circle. 24 evenly-spaced levels from -2.5 to 3.25. Interface —; Negative values —; Positive values —; Exact distance function - -.

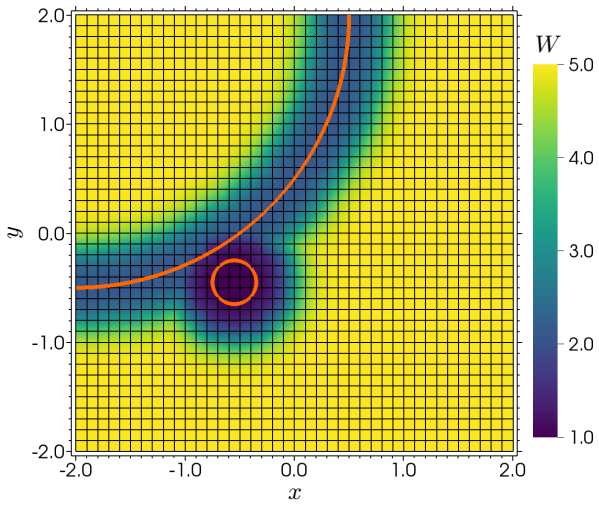


(c) Index function used to define the small circle. Interfaces shown as —.

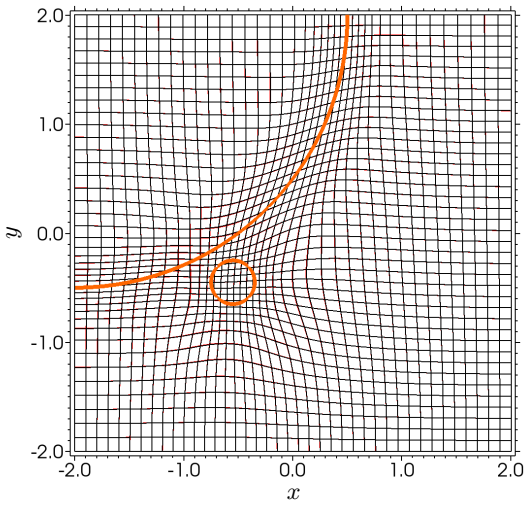


(d) Low-order signed distance function for the small circle. 24 evenly-spaced levels from -2.5 to 3.25. Interface —; Negative values —; Positive values —; Exact distance function - -.

Figure 30: Index function values and low-order signed distance function for the two level sets example.



(a) Weight function computed on initial mesh.



(b) Mesh after 200 iterations. Index function based weights —; Level set based weights —.

Figure 31: Weight function and mesh for the two level sets example using index function based weights. Interfaces are shown as —.

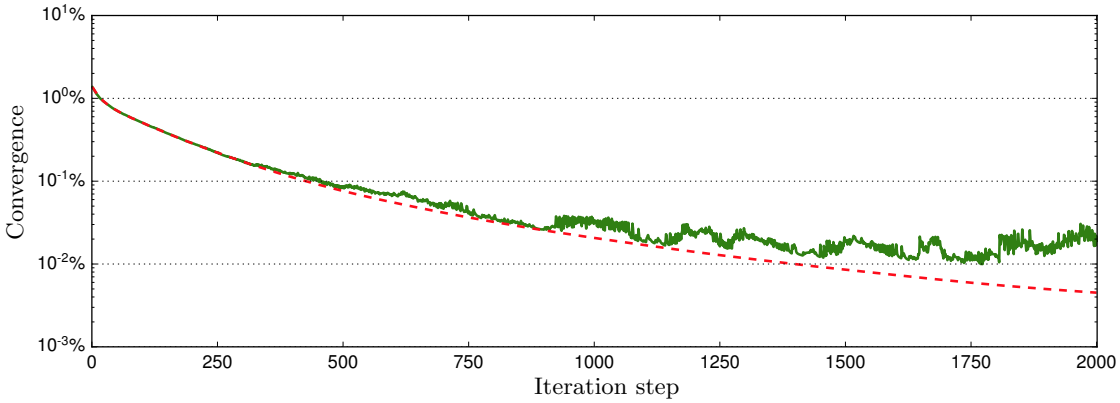


Figure 32: Convergence for the two level sets example using index function based weights. Index function based weights —; Level set based weights - -.

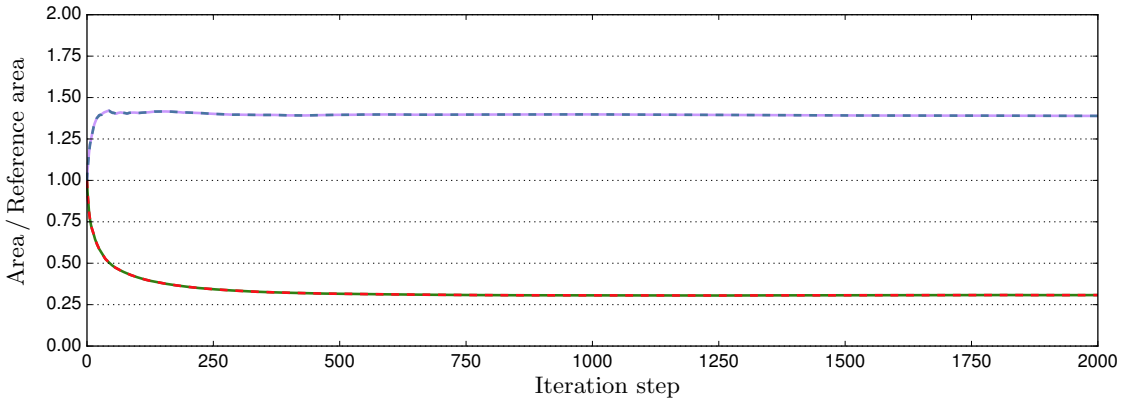


Figure 33: Minimum and maximum cell area for the two level sets example using index function based weights. Index function based minimum area —; Index function based maximum area —; Level set based minimum area - -; Level set based maximum area - -.

generated from the exact signed distance function.

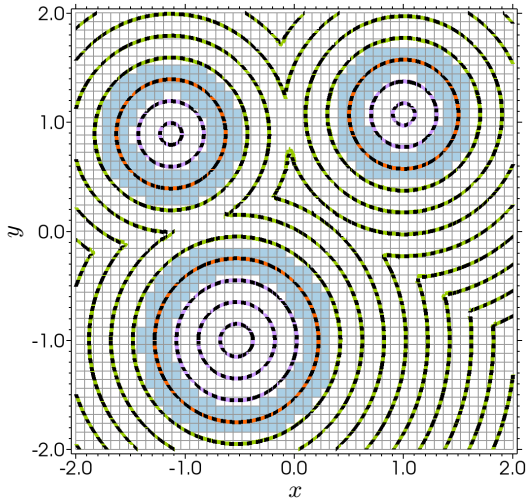
Figure 32 shows the convergence history for this case. Just as in the last example, the mesh motion does not continue to decay. The convergence levels off around 0.02%, again suggesting that the vertices continue to osciallte slightly instead of converging to a final location. Figure 33 shows the minimum and maximum area histories. The values are nearly identical to the level set based values.

5. Combination of distance function and low-order distance function

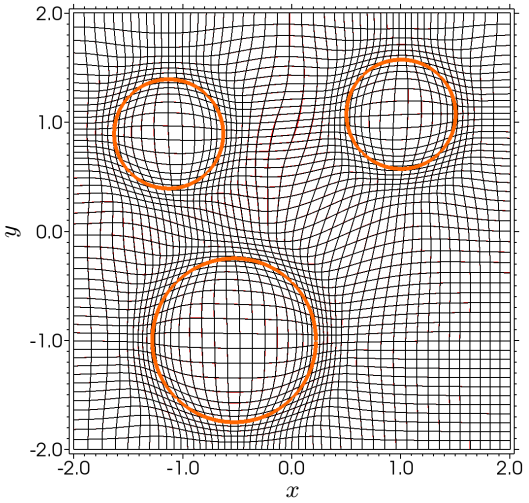
In most level set based codes, the signed distance function is rarely computed over the entire computational domain. Calculations are usually restricted to a small area around the interface. This region is called the "narrow band" in the literature [18]. The size of the narrow band is usually determined by the stencil size of the employed discretization method and is measured in numbers of cells. To use the level set based weights, the signed distance function must be computed from at least d_o^- to d_o^+. This is an issue because the physical distance covered by the narrow band can become only a very small fraction of the distance $|d_o^+ - d_o^-|$ as the mesh is refined.

To address this issue, the narrow-band level set method may be combined with the LOSDF. The level set with its re-distancing method may be used within the narrow band, while outside of the narrow band, the distance function is computed using Equation (4.2) and the rDG(P_0P_2) reconstruction. As an example, the combined sign distance function approach for the three circles case is shown in Figure 34a. In this case, a narrow band of one vertex neighbor from the interface cells is used. In the figure, the narrow band cells are highlighted in blue. For this case, the combined signed distance function does a much better job of estimating the exact distance function, as compared to the LOSDF computed from just the index function. Figure 34b shows the mesh generated from the combined distance function overlapping the mesh created from using just the level set based weights. Unlike the index function case, the combined sign distance function generates a mesh nearly identical to the purely level set based weights.

The convergence history for this example is shown in Figure 35. The pure level set and index function cases are also shown for comparison. The history for the combined distance function weights is nearly identical to the pure level set case. The magnitude of the mesh motion continues to decay, unlike the index function based case. Since the main difference between the combined distance function method and the index function method is the calculation of the first degree of freedom for the interface cells, these results suggest that the source of the oscillations is the inaccuracy of the coarse approximation given by Equation (4.1). The histories for the minimum and maximum cell areas are shown in Figure 36. The results are identical to the level set based results.



(a) Combined signed distance function. 13 evenly-spaced levels from -0.6 to 1.8. Interface —; Negative values —; Positive values —; Exact distance function - -; Narrow band cells ▢.



(b) Mesh after 300 iterations. Combined distance function based weights —; Pure level set weights —.

Figure 34: Signed distance function and mesh for the three circles interface using combined distance function based weights.

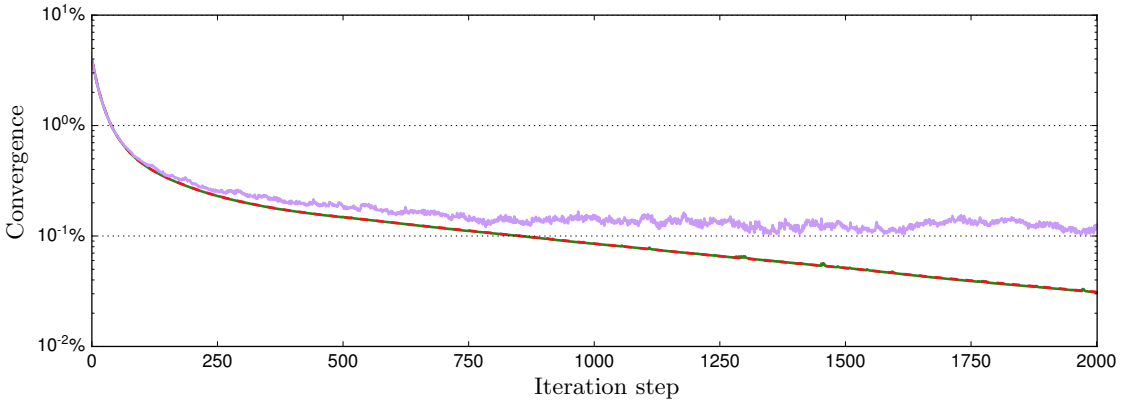


Figure 35: Convergence for the three circles interface using combined distance function based weights. Combined distance function based weights —; Index function based weights —; Pure level set based weights - -.

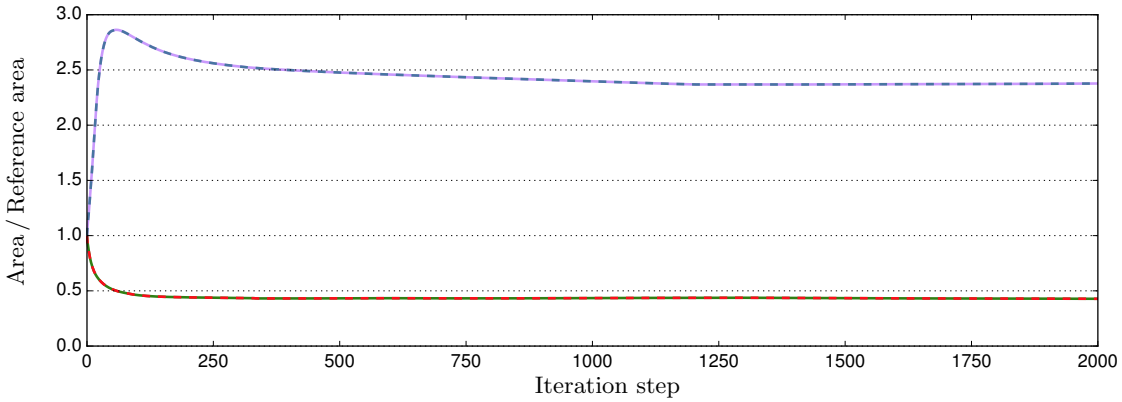


Figure 36: Minimum and maximum cell area for the three circles interface using combined distance function based weights. Combined distance function based minimum area —; Combined distance function based maximum area —; Level set based minimum area - -; Level set based maximum area - -.

6. Specifying a minimum cell area

In all the examples shown so far, the meshes have been able to conform to the specified geometries and provide refinement at the desired locations. The level of refinement in all the examples is controlled by the specified weight function amplitude, A. Due to the global nature of CNR, specifying the same value of A for different geometries can produce different levels of refinement. To address this issue, we now present a method for specifying a final minimum cell area, in place of the weight function amplitude.

The method for specifying a minimum area is composed of two elements: a barrier function that is added to the original functional, and a weight amplitude adjustment procedure. The barrier function takes the form of an exponential that is designed to prevent the area of the mesh cells from becoming significantly smaller than the desired minimum area. With a sufficiently large weight amplitude, the barrier function is all that is needed to enforce a minimum cell area. However, the resulting mesh may oscillate slightly and never converge to a final position. The weight amplitude adjustment procedure is designed to prevent these undesired effects. Once the target cell area is obtained, the weight function amplitude will be reduced whenever the minimum cell area is less than the specified value. In addition, the amplitude will be increased whenever the minimum cell area is much greater than the specified value, which will drive the mesh towards the target area. The goal is to find an amplitude where the minimum area is an acceptable value and the magnitude of the barrier function is approximately zero. The details of the procedure will be presented next, followed by some examples demonstrating the method's ability to obtain the target minimum cell area starting from arbitrary weight function amplitudes.

6.1 Barrier function

The barrier function is simply added to the original functional given in Equation (2.1). The modified functional for vertex p now has the form

$$F(\mathbf{x}_p) = \sum_{c \in \mathcal{C}(p)} \left(W^{cp} \frac{M^{cp}}{J^{cp}} + W^{cp-} \frac{M^{cp-}}{J^{cp-}} + W^{cp+} \frac{M^{cp+}}{J^{cp+}} + B^c \right), \tag{6.1}$$

where B^c is the barrier function for cell c. If Ω_T is the target minimum area, then the barrier function takes the form

$$B^c = A_B \exp\left(-\frac{P_B}{\Delta\Omega_B} (\Omega_T - \Delta\Omega_T - \Omega_c) \right) \tag{6.2}$$

where A_B is the amplitude of the barrier function, $\Delta\Omega_B$ controls the width of the barrier function, P_B controls the strength of the barrier function at the edges specified by $\Delta\Omega_B$, and $\Delta\Omega_T$ specifies how far the barrier function should be from the target area. In this work, A_B is set to 10, P_B is set to 0.03, $\Delta\Omega_B$ is set

to $0.02\,\Omega_R$, and $\Delta\Omega_T$ is set to $0.05\,\Omega_R$. Once the barrier has been added to the functional, the remainder of the mesh relaxation procedure is unchanged.

It is worth noting that for the quadrilateral cell geometry used in this work, the area of cell c can be computed as

$$\Omega_c = \frac{1}{2}\left(J^{cp-} + J^{cp+}\right). \tag{6.3}$$

This also allows the derivatives of the cell area, which are needed for the minimization procedure, to be easily evaluated since the derivatives of J^{cp-} and J^{cp+} were already computed for the original functional.

6.2 Weight amplitude adjustment

The weight amplitude adjustment works by modifying the weight function amplitude whenever the minimum cell area of the mesh is outside a specified bounds. If Ω_m is the minimum cell area for the mesh, then the weight amplitude is decreased whenever $\Omega_m < \Omega_T$. The amplitude is increased whenever $\Omega_m > \Omega_T + \delta\Omega_T$, where $\delta\Omega_T$ is the amount of leeway allowed for the final minimum area. In this work, $\delta\Omega_T$ is set to $0.02\,\Omega_R$. The amplitude is adjusted at the beginning of every relaxation iteration. If A^i is the current amplitude and A^{i+1} is the updated amplitude, A^{i+1} is expressed as

$$A^{i+1} = \begin{cases} \max\left(A^i\sqrt{R_T},\ 0.99A^i\right) & \text{if } \Omega_m < \Omega_T \\ \min\left(A^i\sqrt{R_{\delta T}},\ A^i + 0.25\right) & \text{if } \Omega_m > \Omega_T + \delta\Omega_T \\ A^i & \text{otherwise} \end{cases} \tag{6.4}$$

where

$$R_T = \frac{\Omega_m}{\Omega_T} \qquad \text{and} \qquad R_{\delta T} = \frac{\Omega_m}{\Omega_T + \delta\Omega_T}. \tag{6.5}$$

6.3 Numerical examples

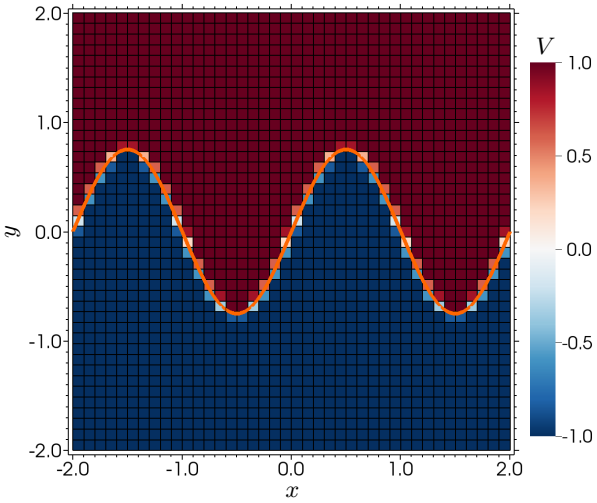
6.3.1 Sinusoidal interface

The first example is the same sinusoidal interface test case defined in Section 3.3.1. Instead of using the exact level set to specified the geometry, an index function will be used this time. Figure 37 shows the index function utilized to define the geometry and the resulting LOSDF. A target minimum cell area of $0.4\,\Omega_R$ was selected for this problem. This is approximately the same minimum area for the mesh generated in Section 3.3.1, so a final weight amplitude of around 5 is expected for the converged value. To demonstrate the versatility of the method, three different initial weight amplitudes (A^0) will be used: 2, 5, and 10. Figure 38 shows the mesh after 1000 iterations for $A^0 = 2$ and $A^0 = 10$. The two cases have converged to nearly identical meshes. The mesh from the $A^0 = 5$ case compares equally well. Table 6 lists the final weight function amplitude, the final minimum mesh cell area, and the final maximum mesh cell area for each of the three cases. All the cases converged to a final weight of approximately 4.69 and the minimum cell area is very close to the target value of $0.4\,\Omega_R$.

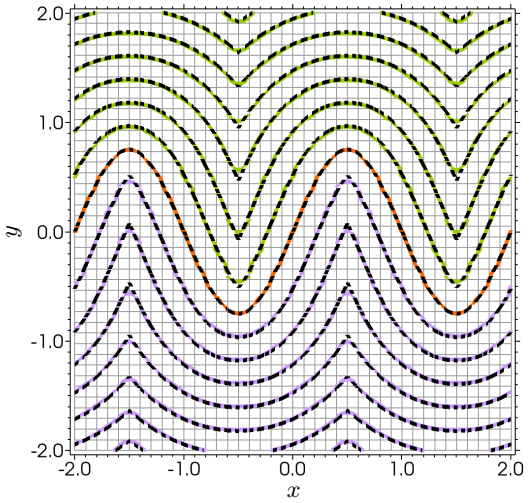
Figure 39 shows the convergence and weight amplitude history for the three cases. All the meshes have very similar convergence histories with the only significant difference observed at the first 50 iterations. The weight amplitude for all three cases are also very similar. They have an initial increase phase, followed by a decrease, and then a small increase again before settling around their final values. All three meshes obtained their final weight amplitude after approximately 250 iterations. Figure 40 shows the minimum and maximum cell area history for all three cases. Similarly to the convergence history, the area histories for each of the meshes follow a very similar trend.

6.3.2 Cruciform interface

The next case demonstrating the method for specifying a minimum weight is the cruciform geometry from Section 4.2.1. For this example, the target minimum area is set to $0.25\,\Omega_R$. Three different initial amplitudes were used: 2, 15, and 30. Figure 41 shows the differences between the $A^0 = 2$ and $A^0 = 30$ cases, and the



(a) Index function. Interface shown as —.



(b) Low-order signed distance function. 15 evenly-space levels from -1.5 to 1.5. Interface —; Negative values —; Positive values —; Exact distance function - -.

Figure 37: Index function and low-order signed distance function for the sinusoidal interface with a specified minimum area.

Initial weight	Final weight	Final minimum area	Final maximum area
2	4.70	0.404	1.72
5	4.69	0.404	1.72
10	4.68	0.405	1.72

Table 6: Weight function amplitude and area values for the sinusoidal interface with a specified minimum area. Area values are non-dimensionalized by reference area.

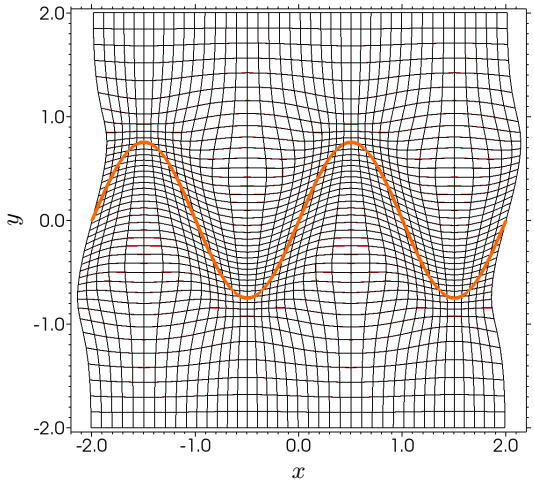
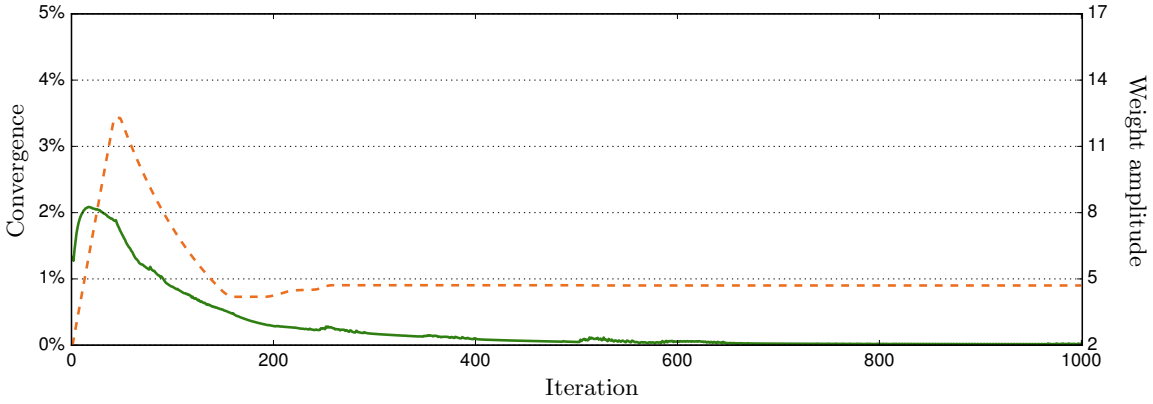
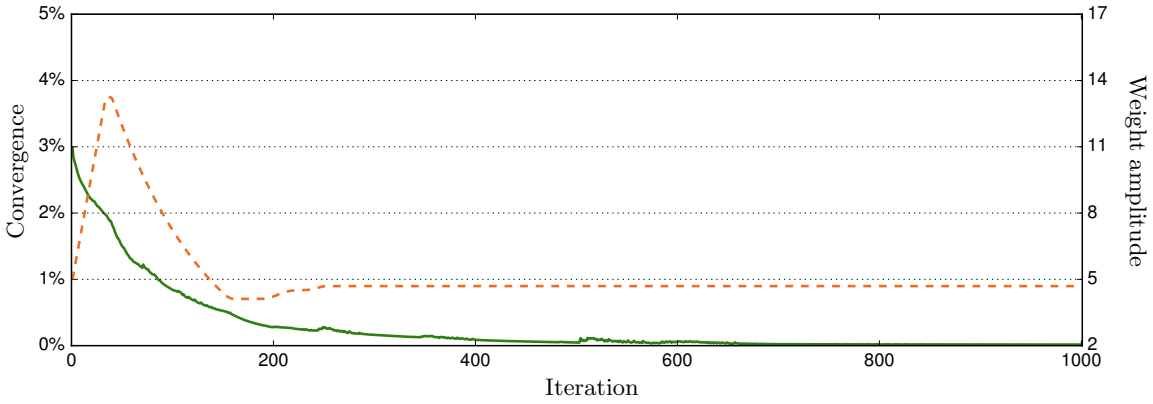


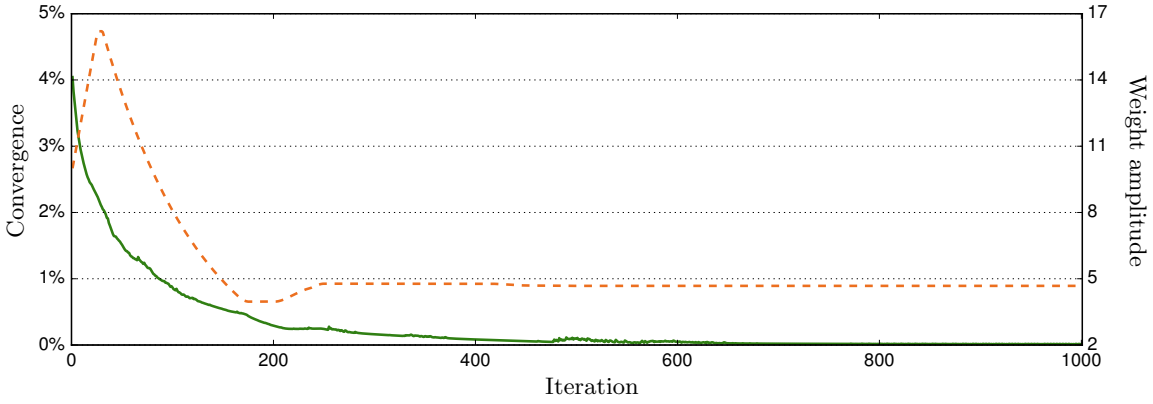
Figure 38: Mesh after 1000 iterations for the sinusoidal interface with a specified minimum area. Mesh with an initial weight of 2 —; Mesh with an initial weight of 10 —.



(a) Initial weight amplitude of 2 case.

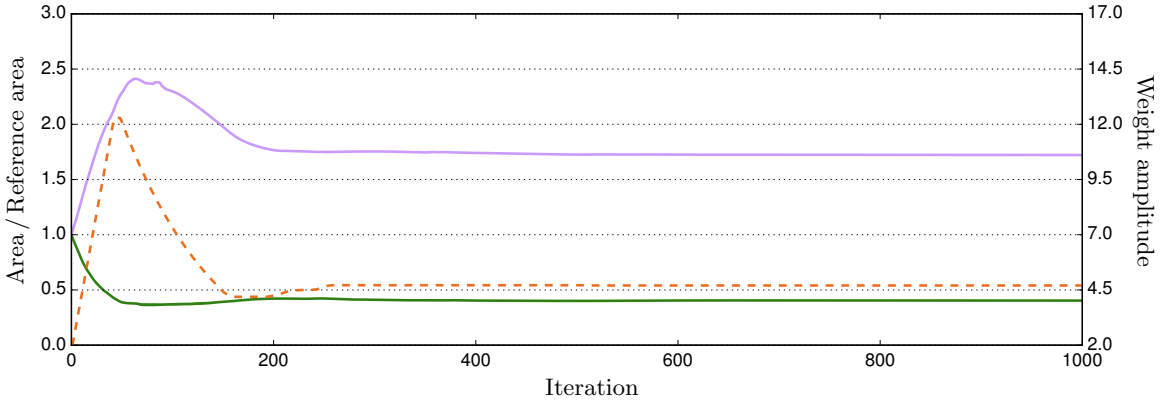


(b) Initial weight amplitude of 5 case.

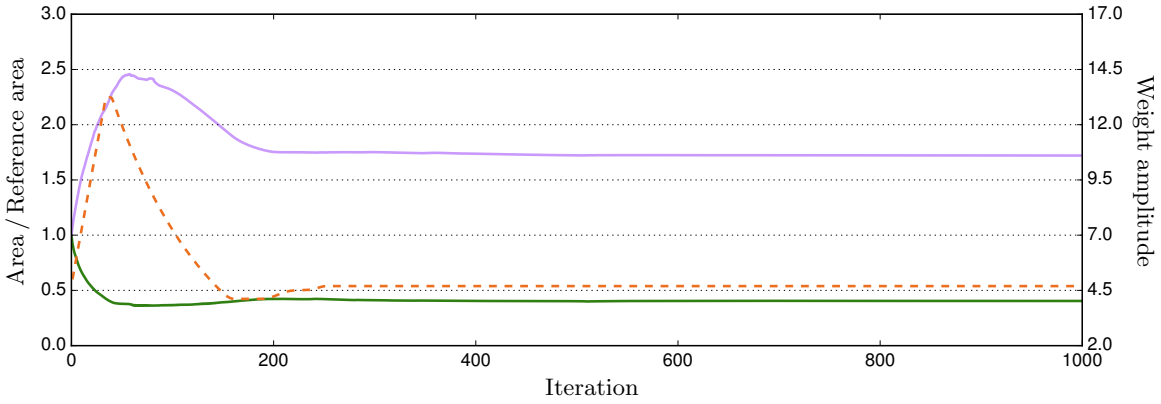


(c) Initial weight amplitude of 10 case.

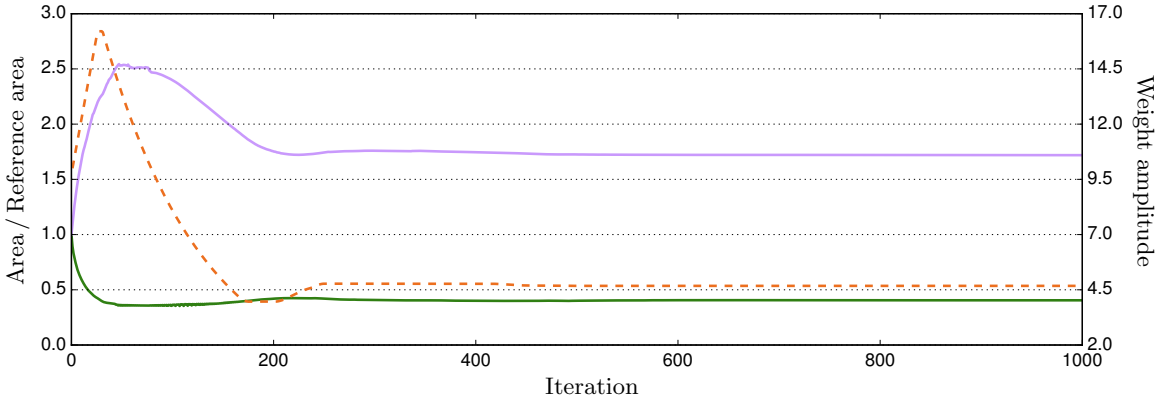
Figure 39: Convergence and weight amplitude for the sinusoidal interface with a specified minimum area. Convergence —; Weight amplitude - -.



(a) Initial weight amplitude of 2 case.



(b) Initial weight amplitude of 5 case.

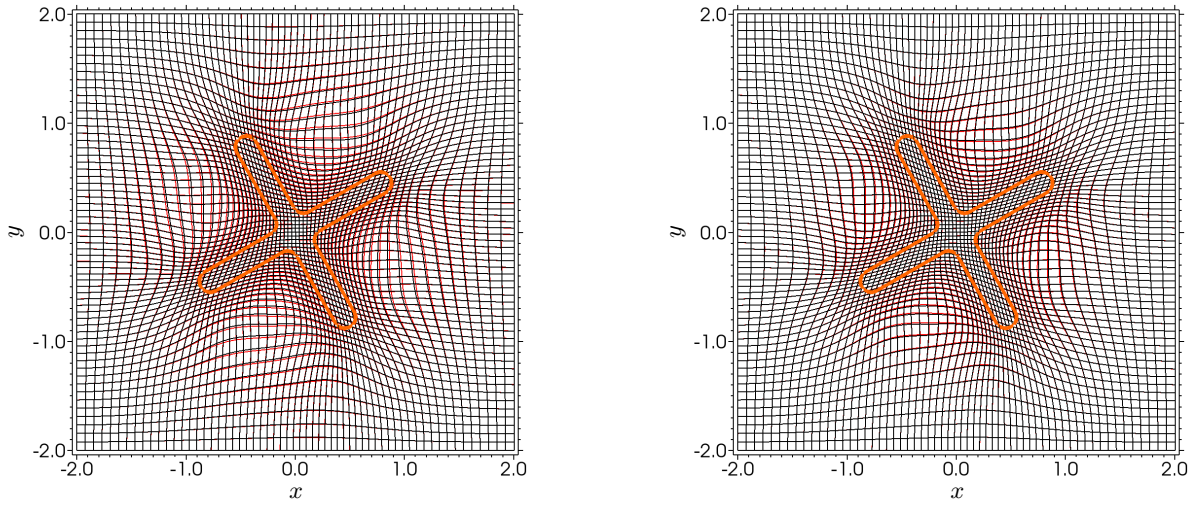


(c) Initial weight amplitude of 10 case.

Figure 40: Minimum cell area, maximum cell area, and weight amplitude for the sinusoidal interface with a specified minimum area. Minimum area —; Maximum area —; Weight amplitude - -.

Initial weight	Final weight	Final minimum area	Final maximum area
2	8.29	0.273	1.91
15	7.42	0.259	2.03
30	6.75	0.262	1.96

Table 7: Weight and area values for the cruciform interface with a specified minimum area. Area values are non-dimensionalized by reference area.



(a) Mesh with an initial weight of 30 —; Mesh with an initial weight of 2 —.

(b) Mesh with an initial weight of 30 —; Mesh with an initial weight of 15 —.

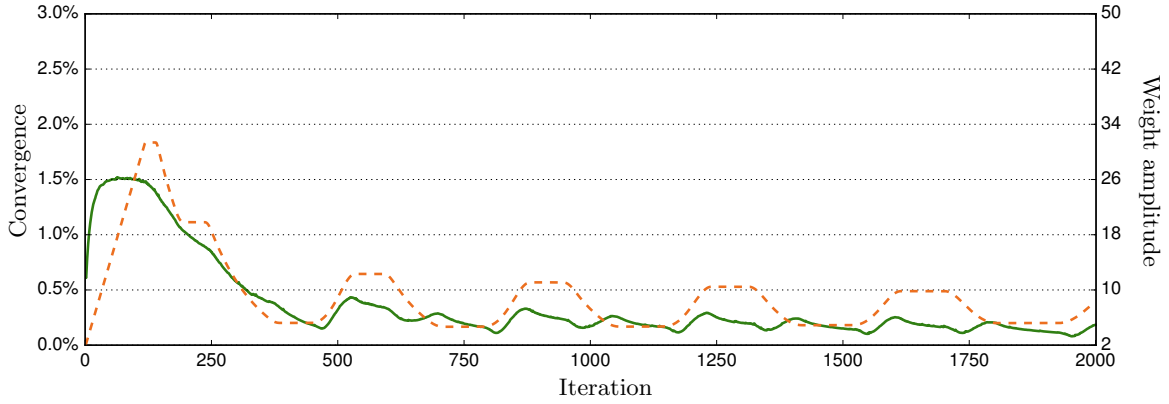
Figure 41: Meshes after 1000 iterations for the cruciform interface with a specified minimum area.

$A^0 = 15$ and $A^0 = 30$ cases, after 1000 iterations. Unlike the sinusoidal example, there are noticeable differences between the meshes. This is due to the fact that the weight amplitude for the $A^0 = 2$ and $A^0 = 15$ cases have not converged yet. Table 7 lists the final weight amplitude, minimum cell area, and maximum cell area for each of the three cases. The results show that the three meshes have clear differences in weight function amplitude and maximum cell area. However, all of them have a minimum cell area close to the target value of $0.25\,\Omega_R$.

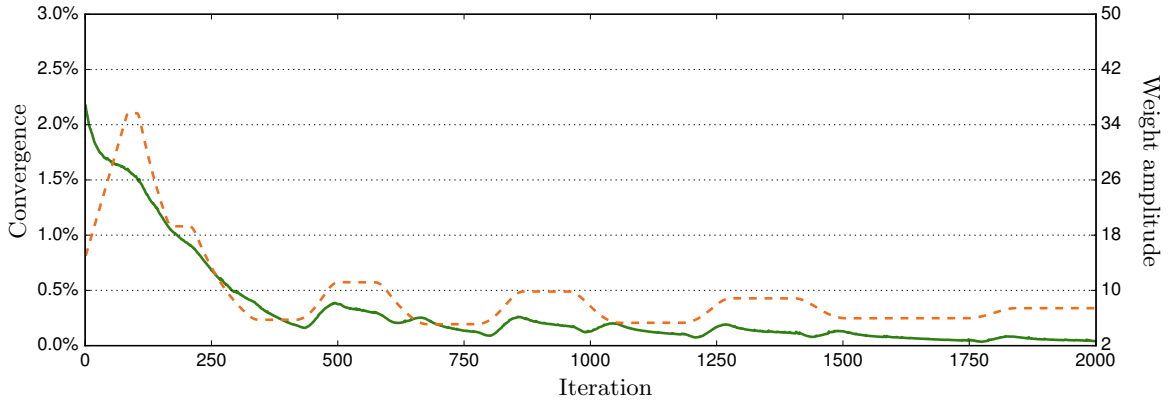
Figure 42 shows the convergence and weight amplitude history for the three cases. The plots clearly show that the $A^0 = 2$ and $A^0 = 15$ cases have not converged after 2000 steps. Although it appears the weight amplitude for the $A^0 = 15$ case is getting close to converging. The $A^0 = 30$ case has converged, as there are no significant changes to the weight amplitude for more than 1000 iterations. The trend seen in the plots suggests that higher initial amplitudes may provided faster convergence. Figure 43 shows the history of the minimum and maximum cell areas. The plots re-enforce the conclusion that the $A^0 = 30$ case has converged, while the $A^0 = 2$ and $A^0 = 15$ cases have not. Despite not converging, the plots show that once the target minimum area was reached, it never diverged from that value.

7. Dynamic Interfaces

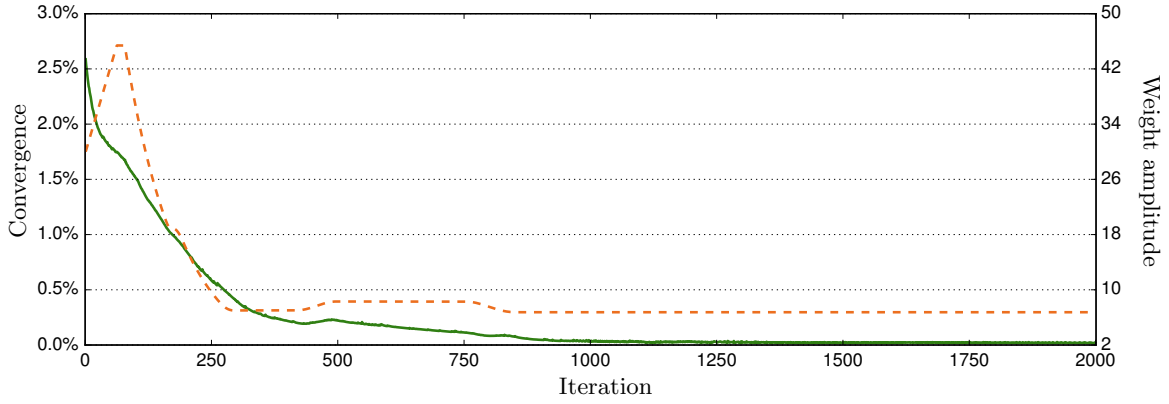
All of the previous examples have involved using mesh relaxation to increase refinement near static interfaces. Those examples can be viewed as a procedure for generating stationary meshes to be used in a purely Eulerian simulation. However, the eventual goal of the weighted condition number mesh relaxation method presented in this work is to be integrated into an ALE framework for use with dynamic interfaces. To demonstrate the potential of the method for ALE simulations, we will show two dynamic interface examples. For both examples, the interfaces will be prescribed by an index function.



(a) Initial weight amplitude of 2 case.

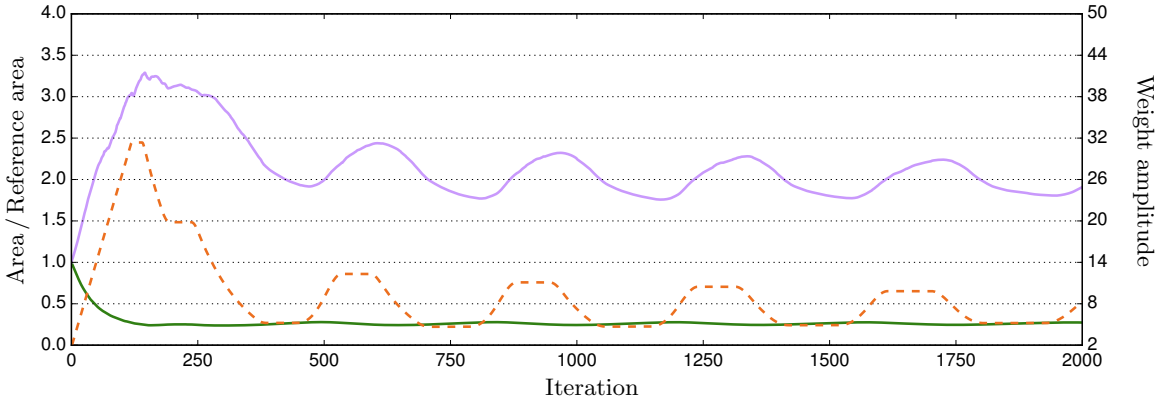


(b) Initial weight amplitude of 15 case.

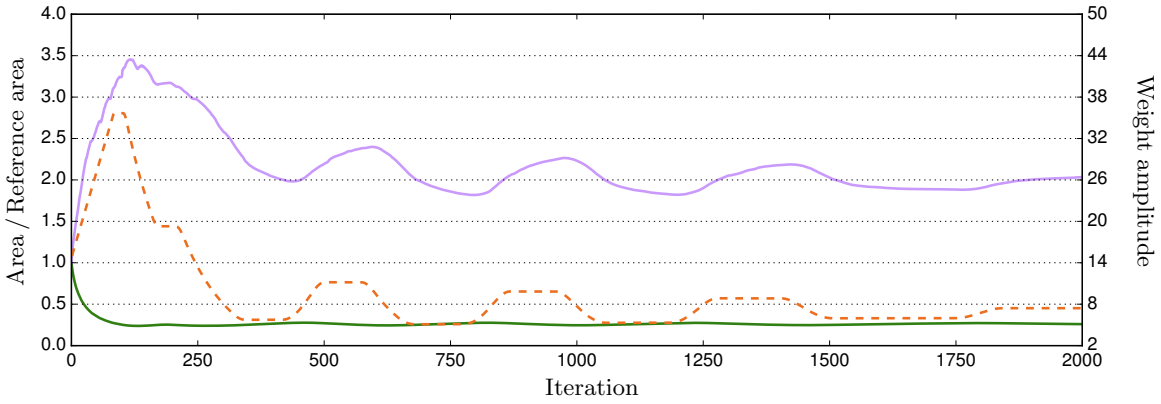


(c) Initial weight amplitude of 30 case.

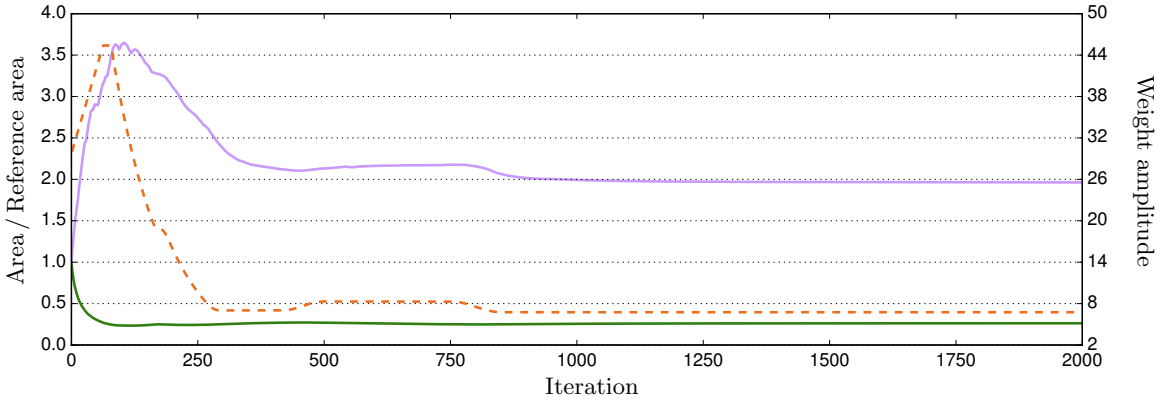
Figure 42: Convergence and weight amplitude for the cruciform interface with a specified minimum area. Convergence —; Weight amplitude - -.



(a) Initial weight amplitude of 2 case.



(b) Initial weight amplitude of 15 case.



(c) Initial weight amplitude of 30 case.

Figure 43: Minimum cell area, maximum cell area, and weight amplitude for the cruciform interface with a specified minimum area. Minimum area —; Maximum area —; Weight amplitude - -.

The dynamic simulations will consist of two steps. First, the interfaces will be kept fixed and the mesh will be relaxed for a number of iterations starting from a uniform mesh. This will generate the initial mesh to be used during the second portion of the simulation. The meshes will be generated with a specified minimum area. In the second part, the interface will be moved with a prescribed motion and a constant CFL number. After each time step, the mesh relaxation method will be employed until a convergence of 0.25% is reached. A minimum of one relaxation step will always be used. During the relaxation, we continue to use the barrier function and weight adjustment process. This is done to allow the weight amplitude to adjust to the dynamics of the interface geometry.

It is worth noting that this is not an exact model of the interface motion found in ALE simulations. In an ALE framework, the mesh will move with the interface during the Lagrangian phase. Once a desired resolution is obtained near the interface during the initial mesh generation, this mesh motion should help keep roughly the same resolution near the interface, which will provided a better initial condition for the mesh relaxation used at each time step. In our dynamic simulations, the mesh is held fixed while the interface is moved. Therefore, the location of refinement from the previous time step can be much further from the interface compared to an ALE simulation. Therefore, one can expect to use more mesh relaxation iteration per time step in our dynamic cases as compared to a similar simulation done in an ALE framework.

7.1 Dynamic sinusoidal interface

The first dynamic example is a "decaying and growing sinusoidal". The initial geometry of the interface and the parameters of the weight function are identical to the static sinusoidal used in Sections 3.3.1 and 6.3.1. When the sinusoidal is set into motion, its amplitude will decay with a non-dimensional velocity of 1. Since the initial magnitude is 0.75, it will take 0.75 time units to reach a magnitude of zero. At that point, the amplitude will increase with a velocity of 1 until the initial amplitude of 0.75 is reached again.

To generate the initial mesh for the dynamic portion of the simulation, a minimum area of $0.4\,\Omega_R$ was specified and the mesh was relaxed with a static interface for 500 iterations. Figure 44 shows the convergence, the weight amplitude, minimum cell area, and maximum cell area histories for the generation of the initial mesh. The plots suggest that the mesh has converged.

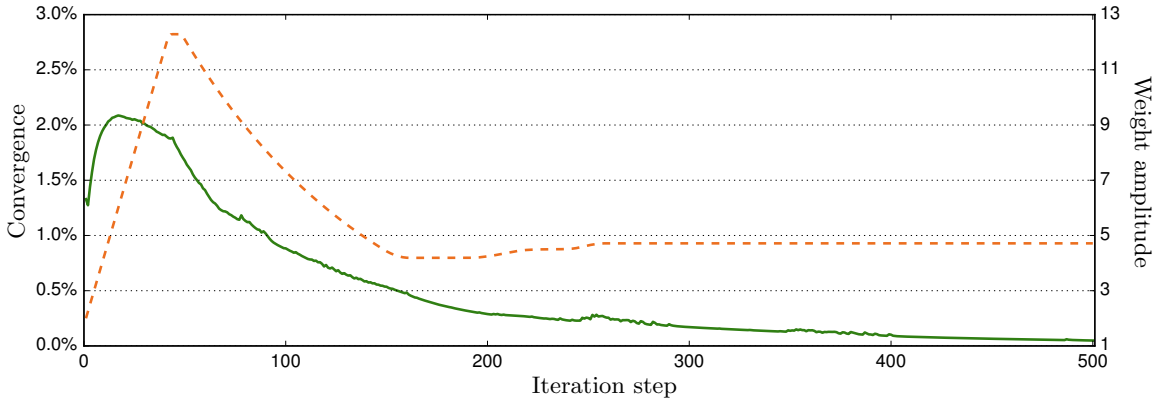
For the dynamic portion of the simulation, three different CFL values were used: 0.1, 0.2, and 0.4. The mesh at different time steps for the CFL = 0.4 case is shown in Figure 45. The figure shows that the mesh relaxation is able to keep pace with the change of the sinusoidal's amplitude. Table 8 provides a comparison of the total number of time steps and the total number of relaxation iterations for each CFL number. When the CFL number is increased from 0.1 to 0.4, the number of required time steps decreases by a factor of 4, as expected, but the total number of relaxation iterations only decreases by a factor of 1.18. This is because the mesh relaxation needs more iterations per time step to adjust to the larger displacement of the interface. Correspondingly, the average number of relaxation iterations per time step increases by a factor of 3.3 when going from a CFL number of 0.1 to 0.4.

Figure 46 shows the number of relaxation iterations for each time step. Also included in the figure is the weight function amplitude at the end of each time step. The figure shows that the weight amplitude changes throughout the simulation. This adjustment is due to the change in the interfacial length. As it gets smaller, a smaller weight function amplitude is need to maintain the prescribed minimum cell area. The figure also shows that changes in the weight amplitude are usually accompanied by spikes in the relaxation iteration count. This is to be expected, as the relaxation must account for not only the change in interface position, but the change in weight amplitude as well. Figure 47 shows the histories of the minimum and maximum cell area for all three cases. Although the maximum cell area changes with the change in weight amplitude, the minimum remains close to the target value for the entire simulation.

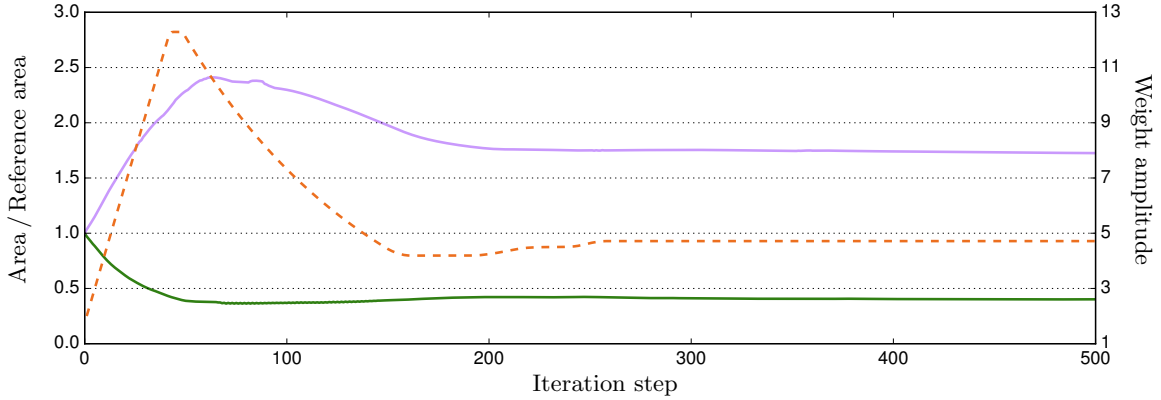
7.2 Dynamic cruciform interface

The second dynamic example is for a rotating cruciform. The initial geometry of the interface and weight function parameters are identical to the previous static cruciform cases, except now the arms are aligned with the x and y axes. The dynamic simulation will cover one full rotation of the cruciform. The period of rotation will have a non-dimensional time of 1.

The initial mesh is generated by relaxing a uniform mesh with a specified minimum area of $0.25\,\Omega_R$ for 1000 iterations. The weight function amplitude was initially set to 2. Figure 44 shows the convergence,



(a) Convergence and weight amplitude. Convergence —; Weight amplitude - -.

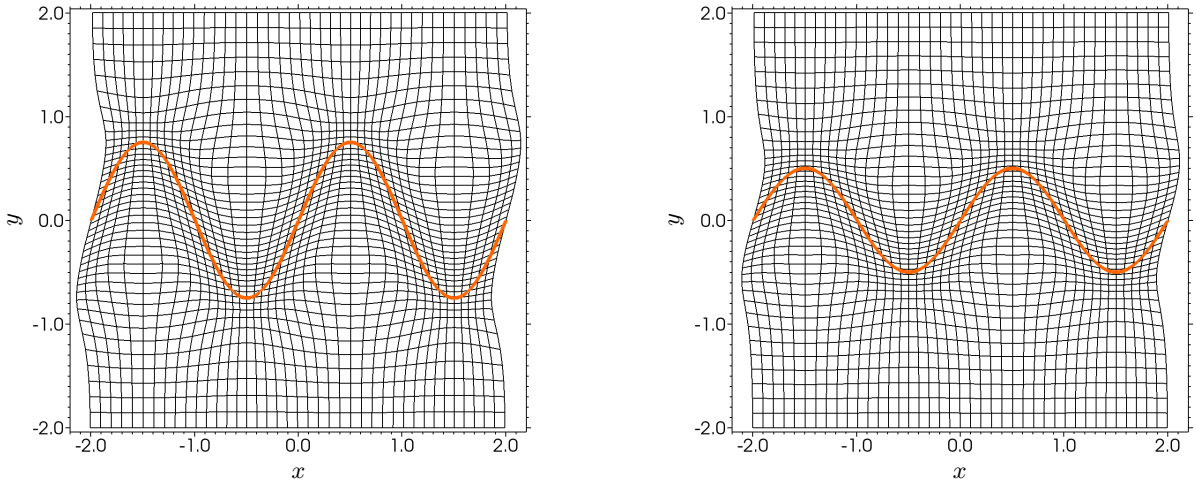


(b) Minimum cell area, maximum cell area, and weight amplitude. Minimum area —; Maximum area —; Weight amplitude - -.

Figure 44: Convergence, weight amplitude, minimum cell area, and maximum cell area for the initial condition to the dynamic sinusoidal interface.

CFL	Total number of time steps	Total number of relaxation iterations	Average number of relaxation iterations per time step
0.1	303	982	3.24
0.2	152	910	5.99
0.4	78	834	10.69

Table 8: Total number of time steps and mesh relaxation iterations for the dynamic sinusoidal interface.



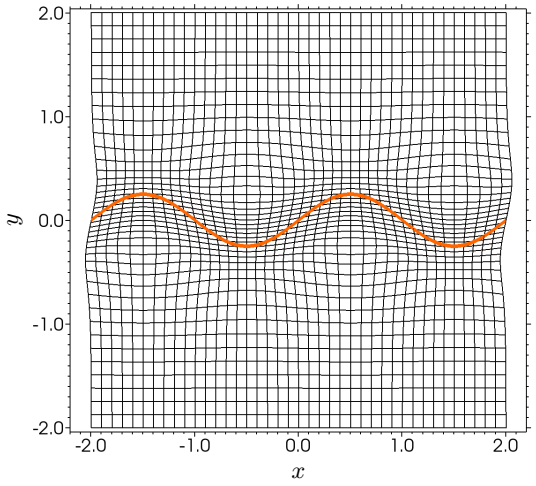
(a) $t = 0.0$

(b) $t = 0.25$

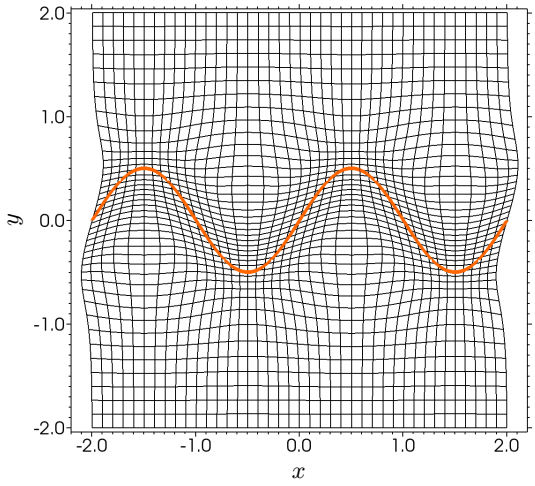
(c) $t = 0.50$

(d) $t = 0.75$

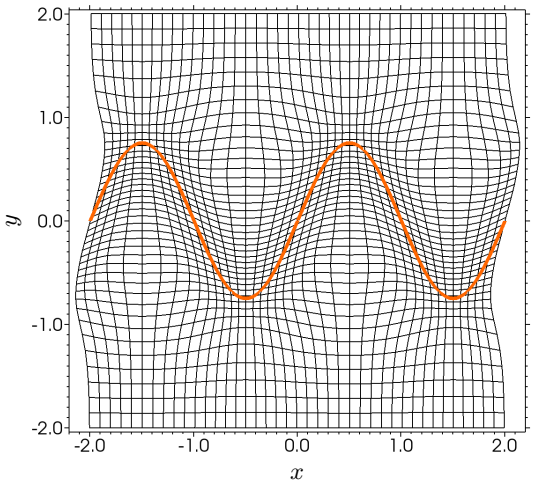
Figure 45: Mesh for the dynamic sinusoidal interface. Mesh is shown at various times during CFL = 0.4 simulation. Interface shown as —.



(e) $t = 1.00$

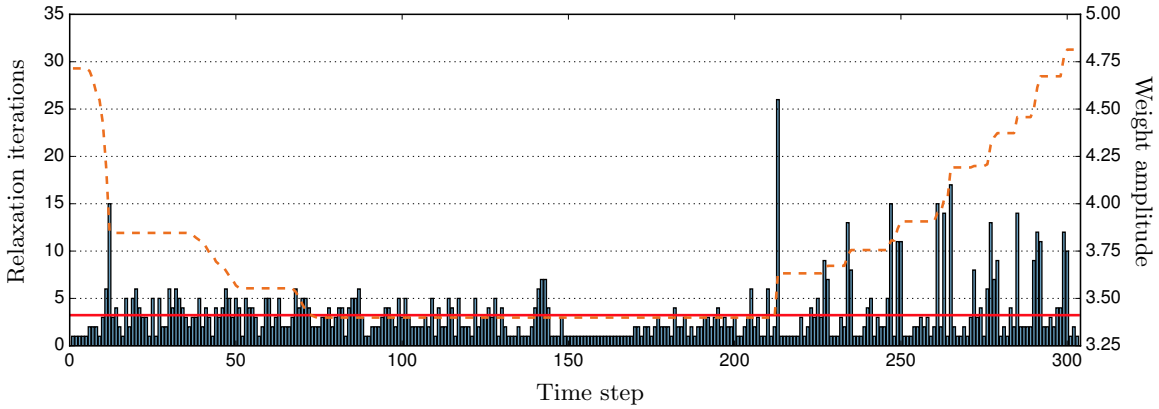


(f) $t = 1.25$

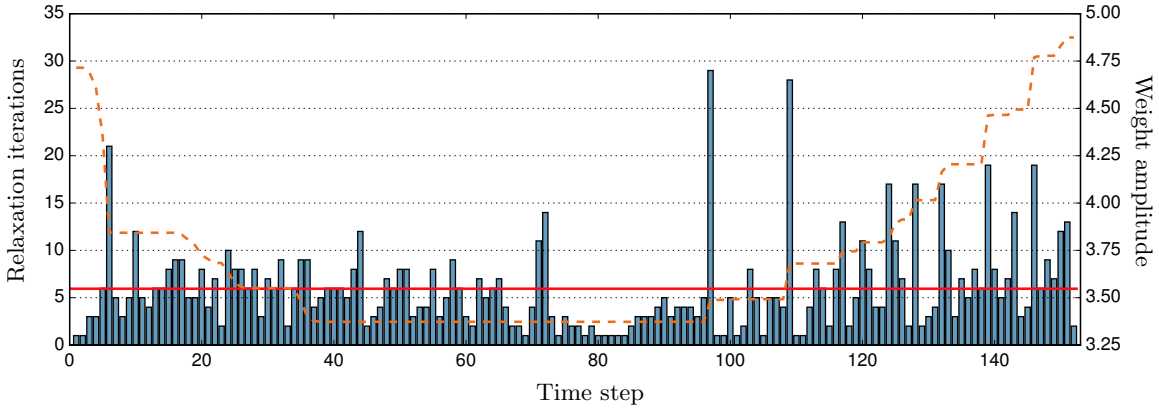


(g) $t = 1.50$

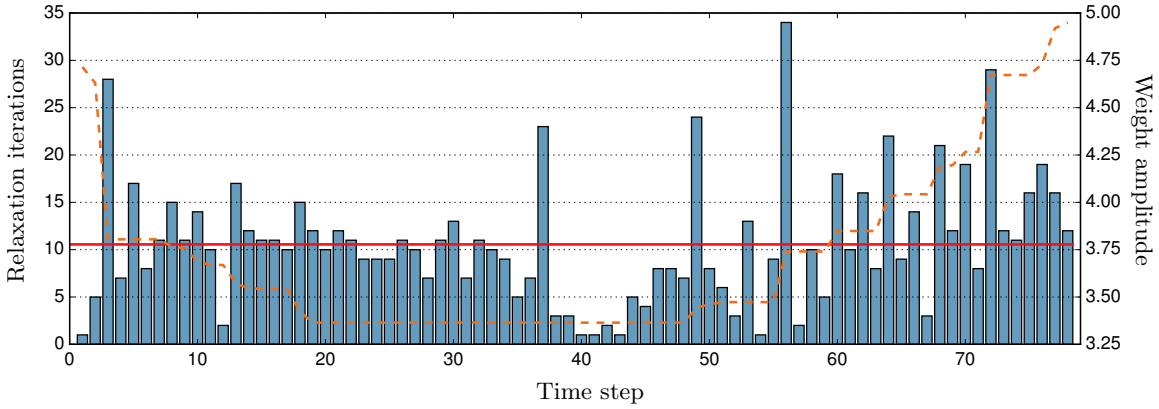
Figure 45 (Continued): Mesh for the dynamic sinusoidal interface. Mesh is shown at various times during CFL = 0.4 simulation. Interface shown as —.



(a) CFL = 0.1 case. Simulation used an average of 3.24 relaxation iterations per time step.

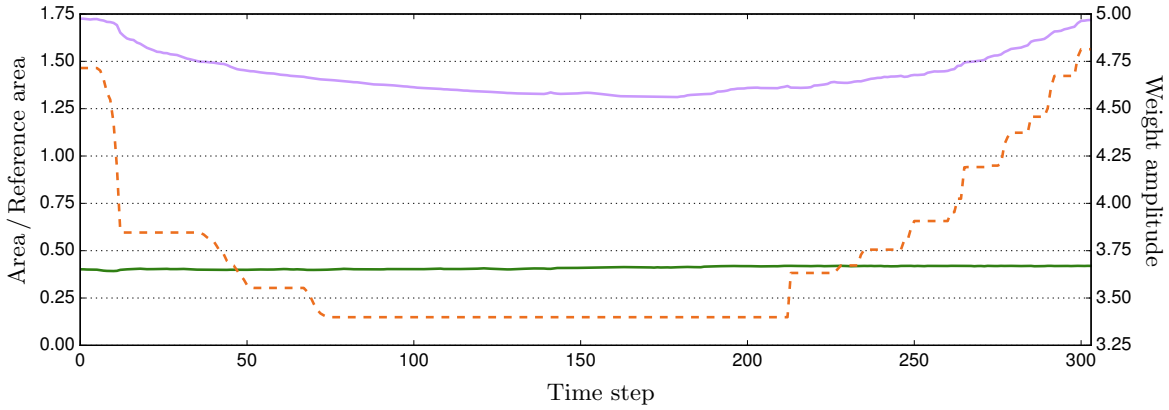


(b) CFL = 0.2 case. Simulation used an average of 5.99 relaxation iterations per time step.

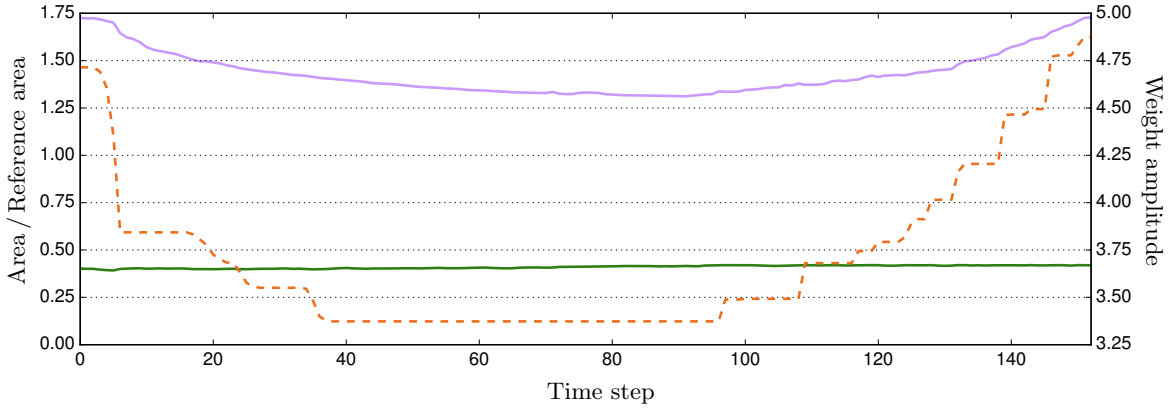


(c) CFL = 0.4 case. Simulation used an average of 10.69 relaxation iterations per time step.

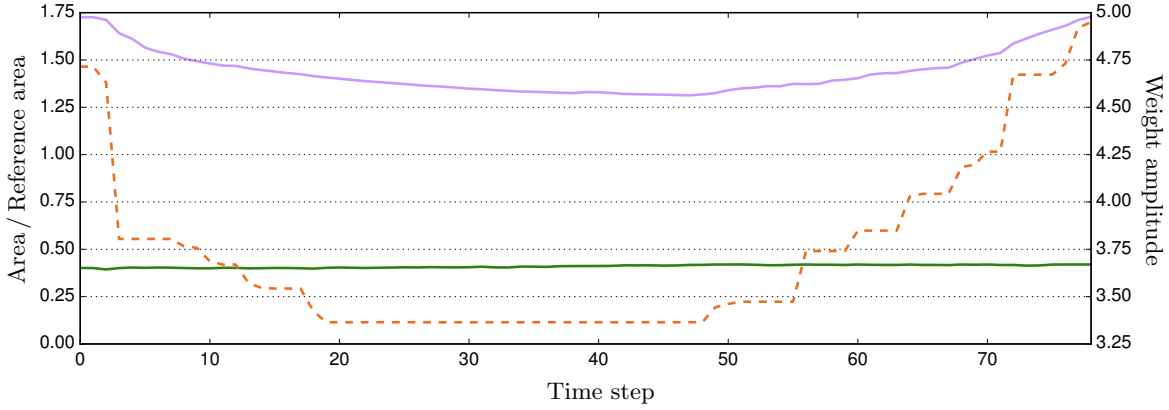
Figure 46: Number of relaxation iterations per time step for the dynamic sinusoidal interface. Number of relaxation iterations per time step ▭; Average number of relaxation iteration per time step —; Weight amplitude at end of time step - -.



(a) CFL = 0.1 case.

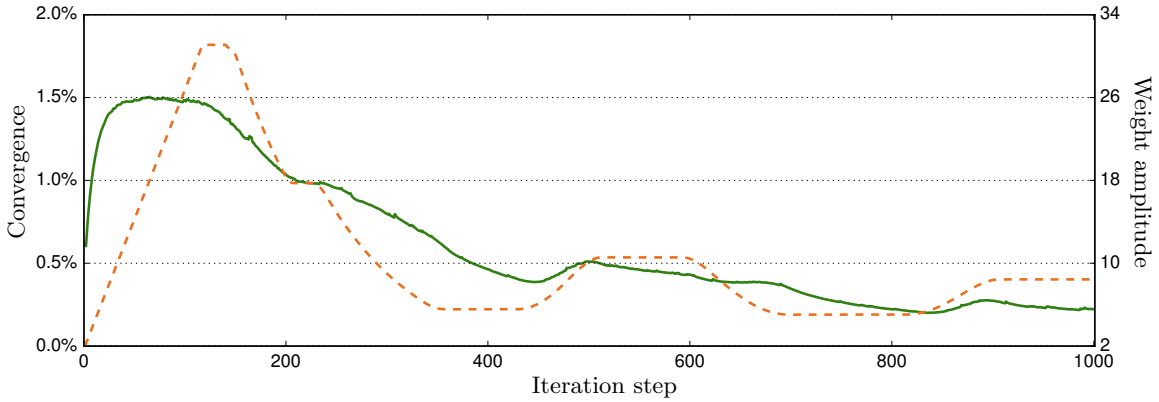


(b) CFL = 0.2 case.

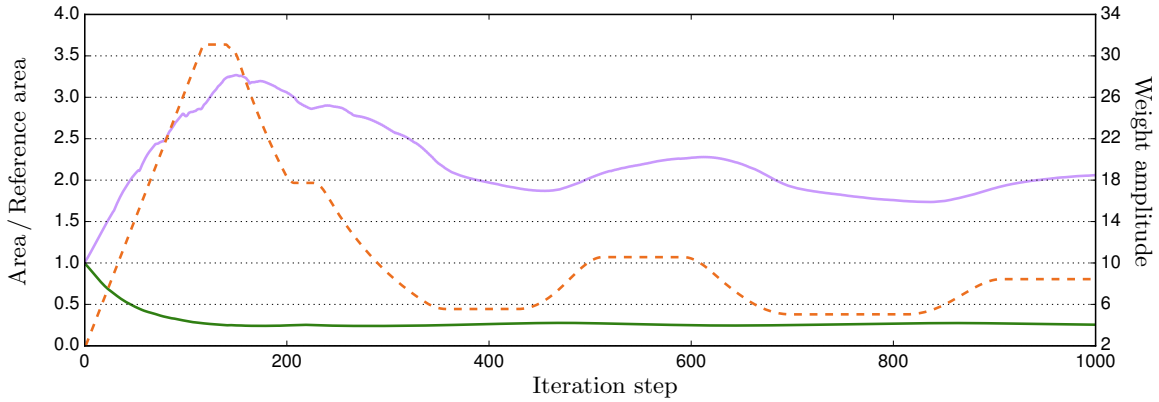


(c) CFL = 0.4 case.

Figure 47: Minimum and maximum cell area for the dynamic sinusoidal interface. Minimum area —; Maximum area —, Weight amplitude - -.



(a) Convergence and weight amplitude. Convergence —; Weight amplitude - -.



(b) Minimum area, maximum area, and weight amplitude. Minimum area —; Maximum area —; Weight amplitude - -.

Figure 48: Convergence, weight amplitude, minimum area, and maximum area for the initial condition to the dynamic cruciform interface.

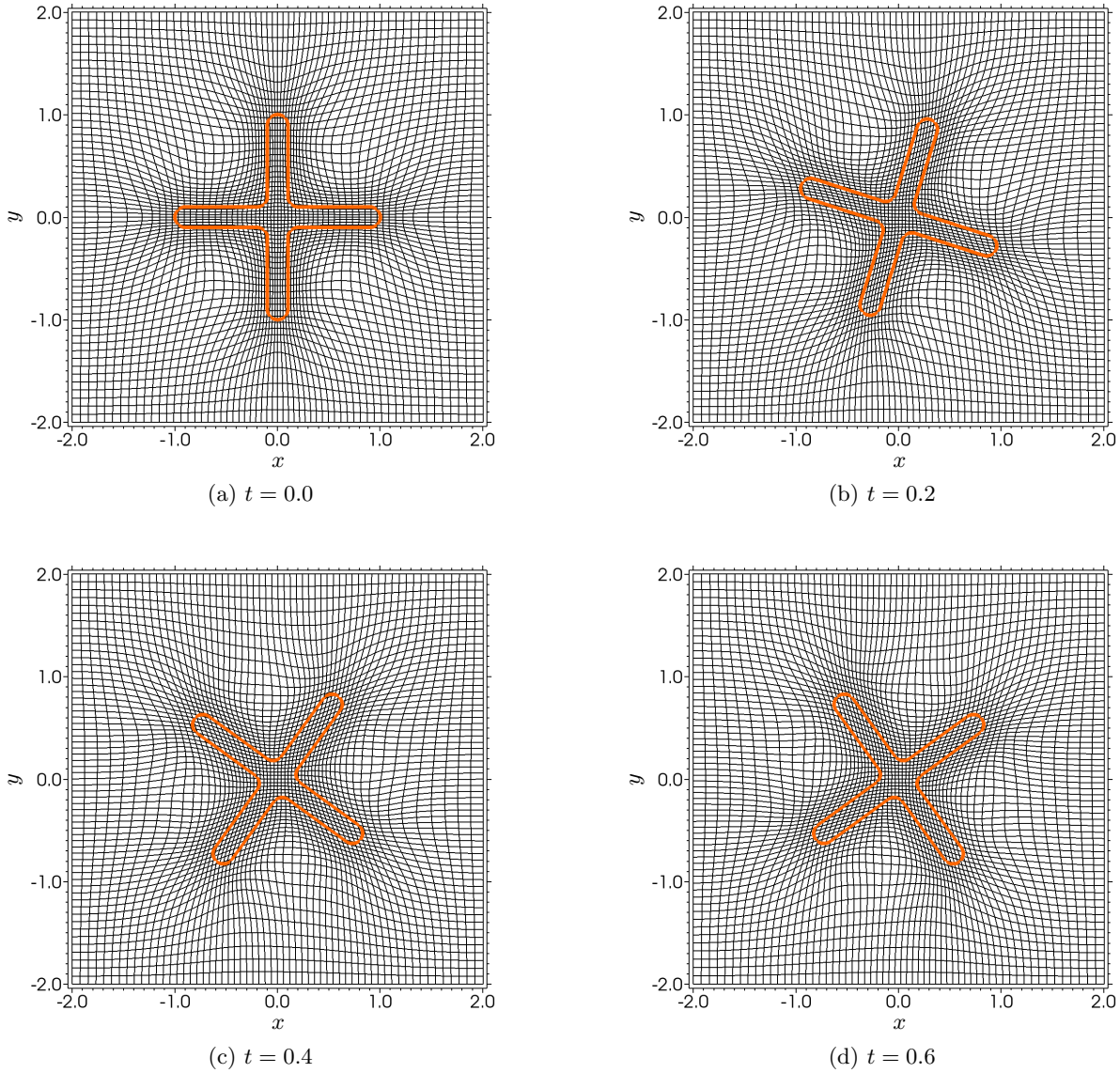
weight amplitude, minimum cell area, and maximum cell area histories for the initial mesh generation. Unlike the dynamic sinusoidal case, the initial mesh has not converged yet. This is done intentionally to demonstrate how the mesh can continue to adjust during the evolution of the interface.

The same three CFL values were used for the dynamic portion of this simulation. The mesh at different time steps for the CFL = 0.4 case is shown in Figure 49. Similarly to the sinusoidal case, the mesh was able to maintain the desired mesh resolution around the cruciform as it rotated. The total number of time steps and total number of relaxation iterations for this case are shown in Table 9. The ratios for the total number of time steps, total number of relaxation iterations, and average relaxation iterations for the CFL = 0.1 and CFL = 0.4 simulations are very similar to the sinusoidal example.

Figure 50 shows the number of relaxation iterations for each time step, up to $t = 0.25$. The figure shows that the weight function amplitude is still able to converge to its final value during the dynamic portion of the simulation. Just as in the sinusoidal example, a change in the weight function amplitude usually comes with a spike in the number of relaxation iterations. Figure 51 is a plot of the minimum and maximum cell area at the end of each time step, covering the entire simulation. The figure shows that the weight function does eventually converge to a final value and remains fixed for a majority of the simulation. The plot also shows that the minimum area remains very close to the target value of $0.25\,\Omega_R$ for the entire simulation.

CFL	Total number of time steps	Total number of relaxation iterations	Average number of relaxation iterations per time step
0.1	1611	7729	4.80
0.2	807	7455	9.24
0.4	405	7039	17.38

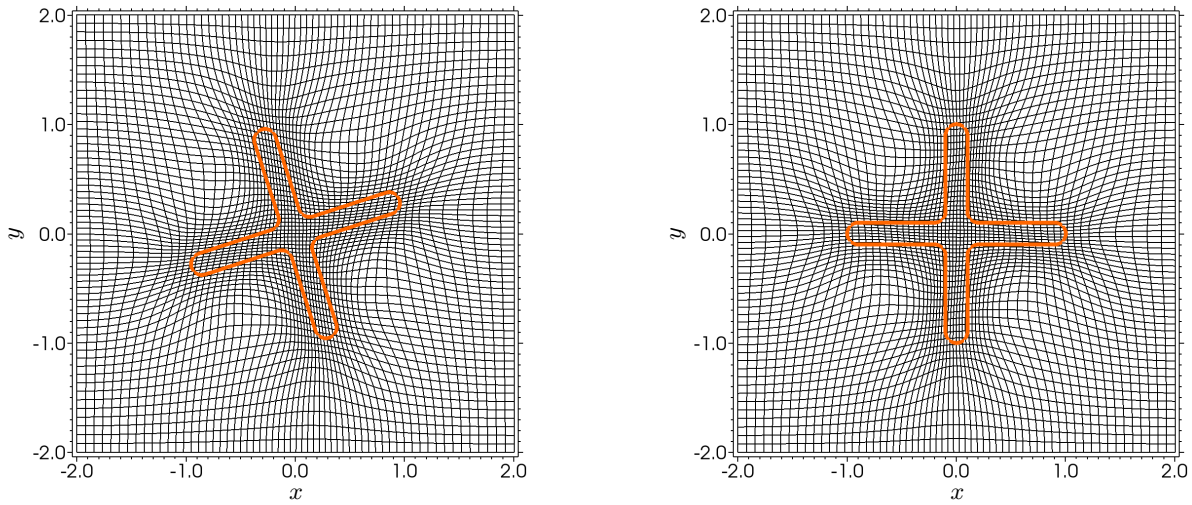
Table 9: Total number of time steps and mesh relaxation iterations for the dynamic cruciform interface.



(a) $t = 0.0$ (b) $t = 0.2$

(c) $t = 0.4$ (d) $t = 0.6$

Figure 49: Mesh for the dynamic cruciform interface. Mesh is shown at various times during CFL = 0.4 simulation. Interface shown as —.



(e) $t = 0.8$ (f) $t = 1.0$

Figure 49 (Continued): Mesh for the dynamic cruciform interface. Mesh is shown at various times during CFL = 0.4 simulation. Interface shown as —.

8. Conclusion

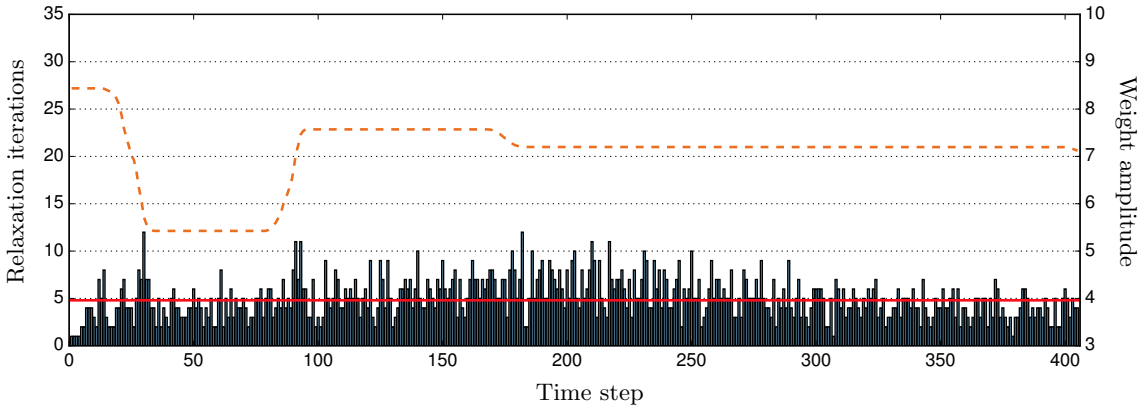
In this work, we presented a new discontinuous Galerkin based weighted condition number mesh relaxation method that can cluster grid cells near dynamically evolving interfaces. The weight function is computed from the interface's level set function and is represented as an rDG(P_1P_2) projection with WENO used to remove in-cell discontinuities. The discontinuous Galerkin projection allows us to readily compute the derivatives required for the condition number optimization. Meshes for a number of example geometries were presented. The examples were chosen to test the method on complex geometries, large differences in interface topology length scales, and multiple materials. For all the cases, the method was able to refine the mesh around the interfaces in the desired fashion.

In many legacy codes, a level set representation of interfaces and fronts may not be available. For these cases, we developed a method for generating a low-order level set function from an interface prescribed by discrete cell-centered values (denoted as an index function in this work). The low-order level set is expressed as an rDG(P_0P_2) projection and is used to compute the weight function in the same manner as the actual level set. Meshes generated using the approximate level set function were shown to be of comparable quality to meshes generated from the exact level set function.

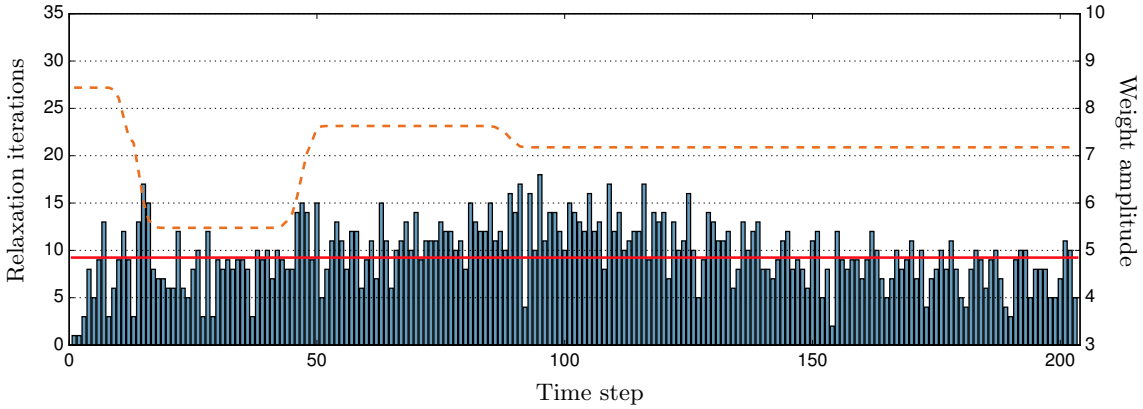
Since most level set based simulations compute the level set only within a narrow band near the interface, the option of combining a level set method in the narrow band, with the low-order level set function outside of the narrow band was explored. This provides a fast and simple way of computing the level set over the entire domain. The mesh generated from this combined level set was found to be nearly identical to the mesh generated from the global level set method.

To provide a more intuitive method for controlling the amplitude of the weight function, we provide a method of prescribing a minimum cell area for the mesh in place of a weight function amplitude. The method is a combination of a barrier function and weight adjustment procedure. The method was shown to converge to a final weight amplitude for static interfaces, while enforcing the specified minimum cell area.

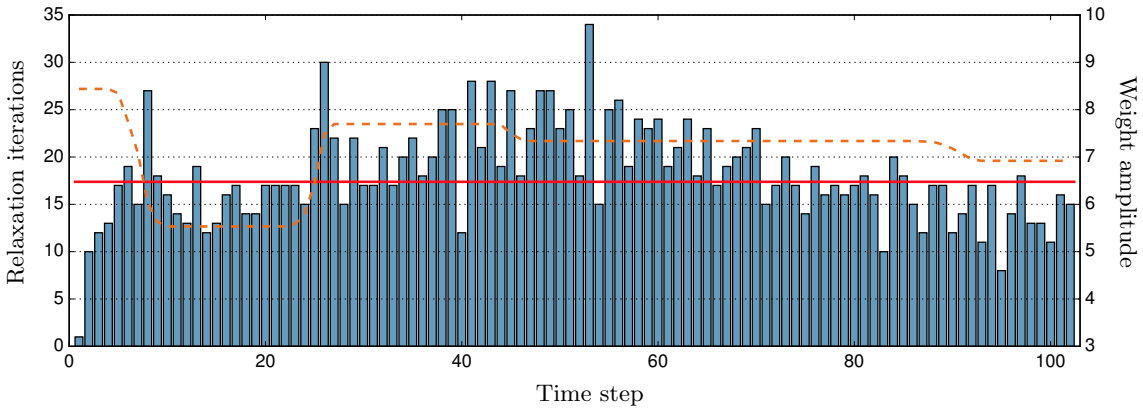
Since the goal of this work is to provide a new mesh relaxation method for ALE simulations, two dynamic interface examples were presented. For both cases, the interface was defined from an index function and a minimum target cell area was specified. The interfaces were moved with a prescribed motion and a fixed CFL number. At each time step, the mesh was allowed to relax until a desired level of convergence was obtained. For both examples, the mesh was able to keep pace with the moving interface and maintain the desired level of resolution, demonstrating the potential usefulness of the method for practical ALE simulations. While the original condition number mesh relaxation used in many ALE methods provides very good mesh



(a) CFL = 0.1 case. Simulation used an average of 4.08 relaxation iterations per time step.

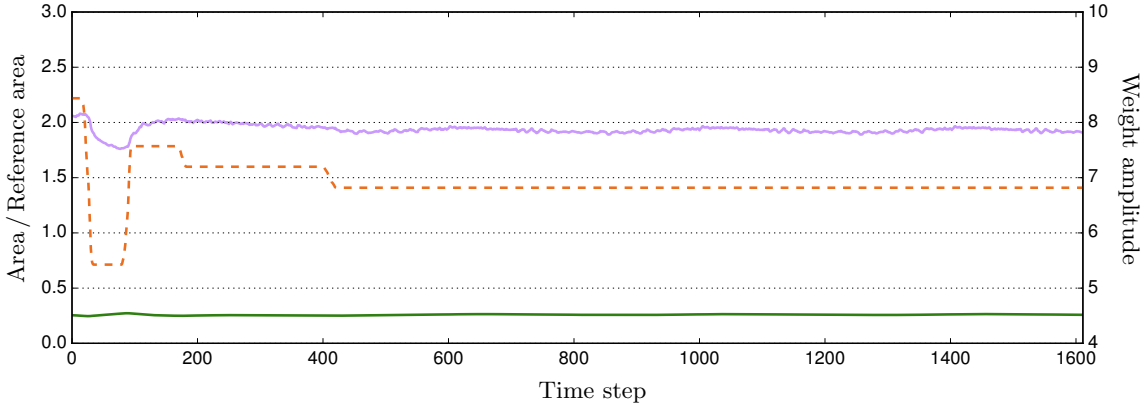


(b) CFL = 0.2 case. Simulation used an average of 9.24 relaxation iterations per time step.

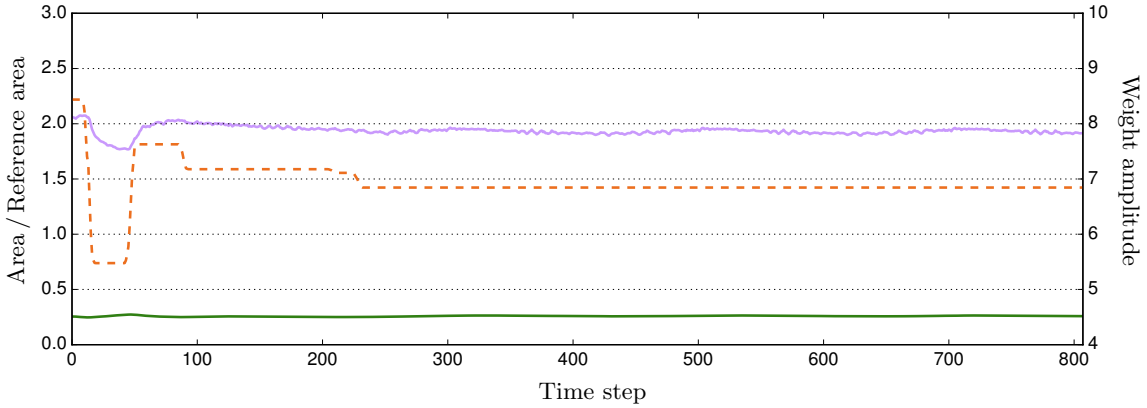


(c) CFL = 0.4 case. Simulation used an average of 17.38 relaxation iterations per time step.

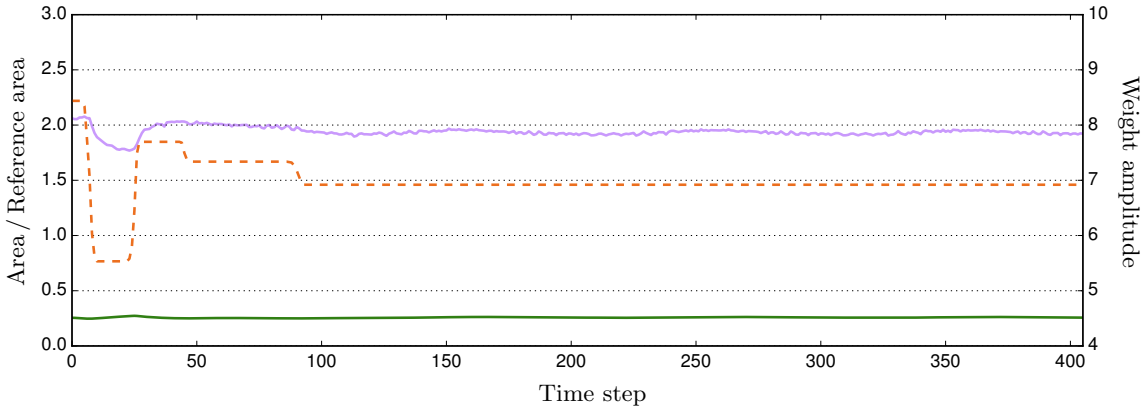
Figure 50: Number of relaxation iterations per time step for the dynamic cruciform interface. Results only shown till a time of 0.25. Number of relaxation iterations per time step ▮; Average number of relaxation iteration per time step —; Weight amplitude at end of time step - -.



(a) CFL = 0.1 case.



(b) CFL = 0.2 case.



(c) CFL = 0.4 case.

Figure 51: Minimum and maximum cell area for the dynamic cruciform interface. Minimum area —; Maximum area —, Weight amplitude - -.

smoothing and prevents mesh tangling, it is unable to maintain the fidelity of the mesh near dynamically evolving multi-material interfaces. The method presented in this work provides an approach for maintaining a desired resolution near a dynamically moving interface, while also retaining the mesh smoothing capabilities of the original condition number mesh relaxation algorithm.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Information management release number LLNL-TN-695525.

Appendix A. WENO reconstruction-based discontinuous Galerkin

For completeness, we provided the details of the WENO reconstruction from Luo et al. [10, 11]. The method begins by computing the constant and linear degrees of freedom from an L_2 projection of the weight function onto the constant and linear basis functions. For a prescribed weight function, $W(\mathbf{x})$, the L_2 projection results in the equation

$$\int_{\Omega_c} W(\mathbf{x})\, b_m^c(\mathbf{x})\, d\Omega = \sum_{n=1}^{N_{P_1}} W_n^c \int_{\Omega_c} b_n^c(\mathbf{x})\, b_m^c(\mathbf{x})\, d\Omega, \tag{A.1}$$

where N_{P_1} is the total number of constant and linear degrees of freedom. For a two-dimensional projection, N_{P_1} is three. By varying m from 1 to N_{P_1}, a set of N_{P_1} equations can be generated. These equations can be written in matrix form as

$$\begin{bmatrix} M_{1,1} & M_{1,2} & M_{1,3} \\ M_{2,1} & M_{2,2} & M_{2,3} \\ M_{3,1} & M_{3,2} & M_{3,3} \end{bmatrix} \begin{bmatrix} W_1^c \\ W_2^c \\ W_3^c \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix}, \tag{A.2}$$

where the mass matrix is

$$M_{i,j} = \int_{\Omega_c} b_i^c(\mathbf{x})\, b_j^c(\mathbf{x})\, d\Omega \qquad \text{and} \qquad R_i = \int_{\Omega_c} W(\mathbf{x})\, b_i^c(\mathbf{x})\, d\Omega. \tag{A.3}$$

Both of the integrals are computed using Gaussian quadrature. With the weight function known at each quadrature point, the constant and linear degrees of freedom (W_1^c, W_2^c, and W_3^c) can be found by solving this system of equations.

With the constant and linear degrees of freedom known, the quadratic terms can be computed using least-squares reconstruction. The reconstruction computes the quadratic values for a given cell from the constant and linear terms in that cell and its face neighbors. This is accomplished by enforcing that the point-wise value and the first derivatives of the reconstructed projection evaluated at each neighboring cell's centroid be equal to the same values computed using the non-reconstructed projection of that neighboring cell. Consider the cell c and its reconstructed projection of the weight function given by Equation (2.4). Let cell j be one of the cells neighboring cell c. Cell j's non-reconstructed projection of the weight function is given as

$$\widetilde{W}^j(\mathbf{x}) = \sum_{n=1}^{N_{P_1}} W_n^j\, b_n^j(\mathbf{x}). \tag{A.4}$$

The non-reconstructed projection is only a function of the constant and linear degrees of freedom. If $\mathbf{x}_c^j$ is the centroid of cell j, then equating the reconstructed projection from cell c to the non-reconstructed projection from cell j gives

$$\begin{aligned} \widetilde{W}^j(\mathbf{x}_c^j) &= W^c(\mathbf{x}_c^j) \\ &= \sum_{n=1}^{N} W_n^c\, b_n^c(\mathbf{x}_c^j). \end{aligned} \tag{A.5}$$

Equating the x derivative gives

$$\begin{aligned}\frac{\partial \widetilde{W}^j}{\partial x}(\mathbf{x}_c^j) &= \frac{\partial W^c}{\partial x}(\mathbf{x}_c^j) \\ &= \sum_{n=1}^{N} W_n^c \frac{\partial b_n^c}{\partial x}(\mathbf{x}_c^j).\end{aligned} \tag{A.6}$$

Equating the y derivative gives

$$\begin{aligned}\frac{\partial \widetilde{W}^j}{\partial y}(\mathbf{x}_c^j) &= \frac{\partial W^c}{\partial y}(\mathbf{x}_c^j) \\ &= \sum_{n=1}^{N} W_n^c \frac{\partial b_n^c}{\partial y}(\mathbf{x}_c^j).\end{aligned} \tag{A.7}$$

These three equations can be written in matrix form as

$$\begin{bmatrix} b_4^c(\mathbf{x}_c^j) & b_5^c(\mathbf{x}_c^j) & b_6^c(\mathbf{x}_c^j) \\ \frac{\partial b_4^c}{\partial x}(\mathbf{x}_c^j) & \frac{\partial b_5^c}{\partial x}(\mathbf{x}_c^j) & \frac{\partial b_6^c}{\partial x}(\mathbf{x}_c^j) \\ \frac{\partial b_4^c}{\partial y}(\mathbf{x}_c^j) & \frac{\partial b_5^c}{\partial y}(\mathbf{x}_c^j) & \frac{\partial b_6^c}{\partial y}(\mathbf{x}_c^j) \end{bmatrix} \begin{bmatrix} W_4^c \\ W_5^c \\ W_6^c \end{bmatrix} = \begin{bmatrix} \widetilde{W}^j(\mathbf{x}_c^j) - \sum_{n=1}^{3} W_n^c\, b_n^c(\mathbf{x}_c^j) \\ \frac{\partial \widetilde{W}^j}{\partial x}(\mathbf{x}_c^j) - \sum_{n=1}^{3} W_n^c \frac{\partial b_n^c}{\partial x}(\mathbf{x}_c^j) \\ \frac{\partial \widetilde{W}^j}{\partial y}(\mathbf{x}_c^j) - \sum_{n=1}^{3} W_n^c \frac{\partial b_n^c}{\partial y}(\mathbf{x}_c^j) \end{bmatrix}. \tag{A.8}$$

Although this provides a closed system of three equations for three unknowns, the same procedure is applied to all the remaining neighbors of cell c with the resulting equations added to this system of equations. This is done to prevent biasing in one direction of the computational domain. For the quadrilateral cells used in this work, cell c will have four face neighboring cells. This will result in a system of 12 equations. The over-determined system is solved in a least-squares sense to obtain the quadratic degrees of freedom for cell c.

At this point, a piece-wise quadratic projection of the weight function is known for each cell. The WENO reconstruction can now be applied to limit the quadratic degrees of freedom. The WENO reconstruction limits the degrees of freedom by computing new values for derivatives at the cell centroids as a convex combinations of derivatives from the current cell and all its face neighbors,

$$\left.\frac{\partial^2 W^c}{\partial x_m\, \partial x_n}\right|_c^{\text{WENO}} = \sum_{k=1}^{5} \omega_k \left.\frac{\partial^2 W^k}{\partial x_m\, \partial x_n}\right|_c, \tag{A.9}$$

where $k = 1$ is the current cell, $k = 2$ to 5 are the face neighbor cells, and ω_k is the WENO weight for cell k. The values of the derivatives at the cell centroids can be easily computed from the degrees of freedom,

$$W_4^c = \left.\frac{\partial^2 W^c}{\partial x^2}\right|_c \Delta x^2, \qquad W_5^c = \left.\frac{\partial^2 W^c}{\partial x \partial y}\right|_c \Delta x \Delta y, \quad \text{and} \quad W_6^c = \left.\frac{\partial^2 W^c}{\partial y^2}\right|_c \Delta y^2. \tag{A.10}$$

The WENO weights are computed as

$$\omega_k = \frac{\tilde{\omega}_k}{\sum_{i=1}^{5} \tilde{\omega}_i}, \tag{A.11}$$

where $\tilde{\omega}_k$ are the non-normalized WENO weights. The non-normalized WENO weights are computed as

$$\tilde{\omega}_k = \frac{\lambda_k}{(\epsilon + o_k)^\gamma}, \tag{A.12}$$

where λ_k is the linear weight for cell k, ϵ is a small positive number to avoid division by zero, γ is a positive integer parameter to control how fast the WENO weights decay for non-smooth projections, and o_k is the smoothness indicator for cell k. In the current work, we set ϵ to 10^{-40}, γ to 2, λ_1 to 1.0, and $\lambda_{k\neq 1}$ to 0.1. For piecewise-quadratic discontinuous Galerkin projections, the smoothness indicator for the quadratic terms is simply,

$$o_k = \left(\left.\frac{\partial^2 W^k}{\partial x_m\, \partial x_n}\right|_c \right)^2. \tag{A.13}$$

Appendix B. rDG(P_0P_2) reconstruction

The rDG(P_0P_2) least-squares reconstruction is very similar to the rDG(P_1P_2) reconstruction used in Section 2.2 (with the details of the reconstruction provided in Appendix A) to compute the weight function projection. However, for this case, only the first degrees of freedom are known so only the point-wise centroid constraint can be used for the reconstruction. We again want to compute the reconstruction in cell c and let cell j be one of the face or vertex neighbors of cell c. Equating the value of the reconstructed projection in cell c evaluated at the centroid of cell j to the value of the non-reconstructed projection in cell j evaluated at the same location gives

$$d_1^j = \sum_{n=1}^{N} d_n^c \, b_n^c(\mathbf{x}_c^j). \tag{B.1}$$

Moving the unknown degrees of freedom to the left side and the known values to the right hand side gives

$$\sum_{n=2}^{N} d_n^c \, b_n^c(\mathbf{x}_c^j) = d_1^j - d_1^c. \tag{B.2}$$

A similar equation can be derived for all the face and vertex neighbors of cell c. If there are a total of J face and vertex neighbors, this will generate a system of J equations, which can be written in matrix form as

$$\begin{bmatrix} b_2^c(\mathbf{x}_c^1) & \cdots & b_N^c(\mathbf{x}_c^1) \\ \vdots & \ddots & \vdots \\ b_2^c(\mathbf{x}_c^J) & \cdots & b_N^c(\mathbf{x}_c^J) \end{bmatrix} \begin{bmatrix} d_2^c \\ \vdots \\ d_N^c \end{bmatrix} = \begin{bmatrix} d_1^1 - d_1^c \\ \vdots \\ d_1^J - d_1^c \end{bmatrix}. \tag{B.3}$$

This system of equations can now be solved using the normal least-squares approach to find the linear and quadratic degrees of freedom in cell c.

References

[1] C. W. Hirt, A. A. Amsden, and J. L. Cook, "Arbitrary Lagrangian-Eulerian computing method for all flow speeds," *Journal of Computational Physics*, vol. 14, no. 3, pp. 227–253, 1974.

[2] P. M. Knupp, L. G. Margolin, and M. Shashkov, "Reference Jacobian optimization-based rezone strategies for arbitrary lagrangian eulerian methods," *Journal of Computational Physics*, vol. 176, no. 1, pp. 93 – 128, 2002.

[3] A. M. Winslow, "Equipotential zoning of two-dimensional meshes," Tech. Rep. UCRL-7312, Lawrence Radiation Laboratory, 1963.

[4] P. M. Knupp, "Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. part i—a framework for surface mesh optimization," *International Journal for Numerical Methods in Engineering*, vol. 48, no. 3, pp. 401–420, 2000.

[5] P. M. Knupp, "Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. part ii—a framework for volume mesh optimization and the condition number of the jacobian matrix," *International Journal for Numerical Methods in Engineering*, vol. 48, no. 8, pp. 1165–1185, 2000.

[6] P. M. Knupp and N. Robidoux, "A framework for variational grid generation: Conditioning the Jacobian matrix with matrix norms," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 2029–2047, 2000.

[7] P. Váchal and P.-H. Maire, "Discretizations for weighted condition number smoothing on general unstructured meshes," *Computers & Fluids*, vol. 46, no. 1, pp. 479 – 485, 2011.

[8] S. P. Schofield, "Weighted condition number relaxation with discrete weights," Tech. Rep. LLNL-PROC-666100, Lawrence Livermore National Laboratory, 2014.

[9] H. Luo, J. D. Baum, and R. Löhner, "A discontinuous Galerkin method based on a Taylor basis for the compressible flows on arbitrary grids," *Journal of Computational Physics*, vol. 227, no. 20, pp. 8875 – 8893, 2008.

[10] H. Luo, Y. Xia, S. Li, R. Nourgaliev, and C. Cai, "A Hermite WENO reconstruction-based discontinuous Galerkin method for the Euler equations on tetrahedral grids," *Journal of Computational Physics*, vol. 231, no. 16, pp. 5489 – 5503, 2012.

[11] H. Luo, Y. Xia, S. Spiegel, R. Nourgaliev, and Z. Jiang, "A reconstructed discontinuous Galerkin method based on a hierarchical WENO reconstruction for compressible flows on tetrahedral grids," *Journal of Computational Physics*, vol. 236, pp. 477 – 492, 2013.

[12] M. Dumbser, D. S. Balsara, E. F. Toro, and C.-D. Munz, "A unified framework for the construction of one-step finite volume and discontinuous Galerkin schemes on unstructured meshes," *Journal of Computational Physics*, vol. 227, no. 18, pp. 8209 – 8253, 2008.

[13] M. Dumbser, "Arbitrary high order $P_N P_M$ schemes on unstructured meshes for the compressible navier–stokes equations," *Computers & Fluids*, vol. 39, no. 1, pp. 60 – 76, 2010.

[14] M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *Journal of Computational Physics*, vol. 114, no. 1, pp. 146 – 159, 1994.

[15] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[16] P. T. Greene, R. Nourgaliev, and S. P. Schofield, "Marker re-distancing (mrd) algorithm for high-fidelity interface tracking on arbitrary meshes," *Journal of Computational Physics*, Manuscript in Preparation.

[17] C. Hirt and B. Nichols, "Volume of fluid (VOF) method for the dynamics of free boundaries," *Journal of Computational Physics*, vol. 39, no. 1, pp. 201 – 225, 1981.

[18] D. Adalsteinsson and J. A. Sethian, "A fast level set method for propagating interfaces," *Journal of Computational Physics*, vol. 118, no. 2, pp. 269 – 277, 1995.